IEEE MICRO

FEBRUARY 1988



The first step toward smart memories: Video RAMs

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC

IEEE MICRO

Volume 8 Number 1 (ISSN 0272-1732) February 1988

Published by the Computer Society



Cover design: Jay Simpson

STAFF

Editor and Publisher
True Seaborn

Assistant Publisher
Douglas Combs

Managing Editor
Marie English

Assistant Editor
Christine Miller

Assistant to the Publisher
Pat Paulsen

Advertising Director
Dawn Peck

Art Director
Jay Simpson

Design and Production
Tricia Hayden

Membership Manager
Christina Champion

Circulation Manager
Paul Zive

Advertising Coordinators
Heidi Rex, Marian Tibayan

Reader Service
Marian Tibayan

Departments

From the Editor-in-Chief 4

MicroView
What's a WISC, anyway? 6

MicroLaw
Reflections on Dirty Harry, dBase 70

MicroStandards
December Board decisions 73

MicroReview
Word, WordPerfect, Macintosh software; buses and links 74

MicroNews
Superconductor update; 32-bit processor benchmarks 76

New Products Design aides;
low-cost, mainframe-performance micros; I/O and storage devices 79

Product Summary 87

Advertiser/Product Index 88

Computer Society Information C3

Change-of-Address form, p. 88; Reader Interest/Service/Subscription cards, p. 88A.

Feature Articles

Video RAMs: Structure and Applications
Jean-Daniel Nicoud

Video RAMs take the first step toward smart memories. Their evolution may follow two directions. 8

GaAs Microprocessors and Digital Systems: An Overview of R&D Efforts
Harry Vlahos and Veljko Milutinovic

Projects based on the emerging technology of gallium arsenide are attracting interest in both scientific and academic communities. 28

The Balance Multiprocessor System
Shreekant Thakkar, Paul Gifford, and Gary Fielland

By pooling 30 VLSI processors and sharing memory, this powerful computer can speed up a standard floating-point program by a factor of 27.4. 57

Coming Next Issue

IEEE Micro's annual Far East issue explores the:

- **Hitachi Gmicro/200 CPU**
- **NEC V70 CPU**
- **Toshiba TX 32-bit CPUs**
- **Tobus architecture**
- **Micro-TRON bus and applications**

Circulation: IEEE MICRO (ISSN 0272-1732) is published bimonthly by the Computer Society of the IEEE: IEEE Headquarters, 345 East 47th St., New York, NY 10017; Computer Society West Coast Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578. Annual subscription: $18 in addition to Computer Society or any other IEEE society member dues; $27 for members of other technical organizations. Single copies: $10 for members; $20 for nonmembers. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to IEEE MICRO, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578. Second class postage is paid at New York, NY, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons: those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Reprints, IEEE MICRO, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578. All rights reserved. Copyright © 1988 by the Institute of Electrical and Electronics Engineers, Inc.

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services in New Products, Product Summary, MicroReview, MicroNews, MicroView, and MicroLaw reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the Computer Society of the IEEE. Send editorial correspondence to IEEE MICRO, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578. All submissions are subject to editing for style, clarity, and space considerations. IEEE MICRO subscribes to The Computer Press Association's code of professional ethics.

BPA

Wescon/88®

CALL FOR SESSIONS

Sponsored by:
Los Angeles and San Francisco Bay Area Councils, IEEE

Southern and Northern California Chapters, ERA

WESCON/88 PROFESSIONAL PROGRAM

November 15-17, 1988
Anaheim, California

The Wescon/88 Professional Program Committee invites you to propose a Professional Session for Wescon/88 to be held in Anaheim on November 15-17, 1988.

Wescon is a broad based show and convention covering the breadth of the electronics industry. Professional Sessions emphasize current or near-term applications, advances and trends which engineers should consider for present practice in product design and manufacture. A two-hour session presents a collection of complementary papers on a topic of broad interest to the electronics engineering community.

GOAL — The Wescon Professional Program is designed as a vehicle for the continuing education of the technical professional. Its goal is to help the individual advance within the electronic industry by enhancing his or her knowledge of advances in a broad spectrum of topics.

FOCUS — Advances in Electronics for Automation, Superconductivity, Neural Networks, and Computer Systems Advance Applications

Although Wescon/88 will focus on the special topics noted above it is the intention of the Professional Program Committee, in keeping with the stated goal, to offer sessions covering a wide range of topics. Prospective session organizers are encouraged to submit proposals on a variety of subjects of interest to the technical community.

CALL FOR SESSION PROPOSALS — A partial list of areas of interest is shown below

- Communications Systems
- Computer Sciences
- Engineering Workstations
- Component/Device Technology
- Micro/Millimeter Waves
- Manufacturing Technology
- Lightwave Communications
- Local Area/Local Loop Networks
- Instrumentation and Measurement
- Circuit and System Design
- Electromagnetic Compatibility
- Signal Processing

In addition, the Wescon/88 Professional Program Committee will also entertain proposals in related areas such as Marketing, Business and Finance, and Technology and the Law.

SESSION PROPOSAL DEADLINE DATE: MARCH 7, 1988

PLEASE POST AND CIRCULATE

This Call is for Session proposals, not for individual papers. To be selected, a Session proposal must consist of a minimum of three and a maximum of five individual papers to be presented in the two hour time limit, including time for questions from the floor. The Session topic should meet the requirements described above.

In addition to oral presentations, Session papers are printed in two formats. Complete typewritten manuscripts for each individual paper in the selected Sessions must be submitted in advance to meet the publication deadlines. The Session Record includes the three to five papers presented in the Professional Session and is available during the convention. The Wescon/88 Conference Record, a hard-bound volume containing all presented papers, is distributed following the convention to technical libraries throughout the world. Authors-speakers whose manuscripts are not received by the established deadlines will be withdrawn from the program.

The deadline for receipt of Session proposals is **Monday, March 7, 1988.**

Notifications of preliminary acceptance will be mailed on or about the **week of April 4, 1988.** Enclosed will be a Progress Report Form on which you must submit a more detailed report on the progress and development of your session by **Monday, May 9, 1988.** If the Progress Report Form is satisfactory, final acceptance will be announced by the Program Committee by **June 20, 1988,** and detailed instructions for manuscript preparation will be sent to all participants.

Target Date for Receipt of Manuscripts:
Monday, August 1, 1988
Absolute Deadline for Receipt of Manuscripts:
Monday, August 15, 1988

Author-speakers whose manuscripts are not received by the deadline will not be allowed to participate in the Wescon/88 Professional Program.

For a Session proposal form contact:

Wescon/88 Professional Program Committee
c/o Electronic Conventions Management
Educational Activities Department
8110 Airport Boulevard, Los Angeles, CA 90045
213-772-2965
Outside California (800) 421-6816
Inside California (800) 262-4208

Wescon/88 PROFESSIONAL PROGRAM COMMITTEE

Chairman:
William E. Pastori, EATON Corporation
Electronic Instrumentation Division, Los Angeles, CA

Vice Chairpersons:
Dennis Leitterman
Hewlett Packard
Cupertino, CA

James J. Farrell III**
VLSI Technology Inc.
Tempe, AZ

Members:

Jim Flournoy***
Procedamas
Rosemead, CA

Gary Gray
Orange County
Communications
Orange, CA

Jay Kamdar
National Semiconductor
Santa Clara, CA

Robert A. King**
Motorola Inc.
Austin, TX

M. Wayne Murray
UNISYS
Mission Viejo, CA

True Seaborn
IEEE Computer Society
Los Alamitos, CA

Ex-Officio:
Chairman Emeritus:
Byron E. Thinger
Pacific Gas and Electric Company
San Francisco, CA

Convention Director
William E. Murray
Douglas Aircraft Co.
Long Beach, CA

Show Director
Roger Sandmeier
O'Donnell Associates
Torrance, CA

Chairman of the Board
Howard Christie
Canoga Park, CA

**Spotlight Topic Sub-Committee
***Computer Systems Advance Applications

IEEE Micro

The Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
(714) 821-8380

Editor-in-Chief:
James J. Farrell III
VLSI Technology Incorporated*

Associate Editor-in-Chief:
Joe Hootman
University of North Dakota

Editorial Board
Shmuel Ben-Yaakov
Ben Gurion University of the Negev
Dante Del Corso
Politecnico di Torino, Italy
John Crawford
Intel Corporation
Stephen A. Dyer
Kansas State University
K.-E. Grosspietsch
GMD, Germany
David B. Gustavson
Stanford Linear Accelerator Center
Victor K.L. Huang
AT&T Information Systems
Barry W. Johnson
University of Virginia
David K. Kahaner
National Bureau of Standards
Jay Kamdar
National Semiconductor Corporation
G. Jack Lipovski
University of Texas
Kenneth Majithia
IBM Corporation
Richard Mateosian
Marlin H. Mickle
University of Pittsburgh
Varish Panigrahi
Digital Equipment Corporation
Ken Sakamura
University of Tokyo
Michael Smolin
Smolin & Associates
Richard H. Stern
Yoichi Yano
NEC Corporation

Magazine Advisory Committee
Sallie Sheppard (chair), Sumit DasGupta, Michael Evangelist, James J. Farrell III, Ted Lewis, David Pessel, H.T. Seaborn, Bruce D. Shriver, John Staudhammer

Publications Board
James H. Aylor (chair), Charles B. Silio (vice chair), Victor Basili, Richard Burke, Jon T. Butler, J. Thomas Cain, Sumit DasGupta, Michael Evangelist, James J. Farrell III, Glen G. Langdon, Ted Lewis, Ming T. Liu, Theo Pavlidis, David Pessel, Sallie Sheppard, Bruce D. Shriver, John Staudhammer, Harold Stone, Steven L. Tanimoto

*Submit six copies of all articles and special-issue proposals to James J. Farrell III, 8375 South River Parkway, Tempe, AZ 85284; (602) 752-6222; Compmail+ j.farrell.

Department

From the Editor-in-Chief

1988

Our new year seems to be one of those "shaky" times for the people that predict the economy. They give the computer and semiconductor industry moderate projections, at best. Their consensus seems to be one of "slow growth" during this year.

The extremely weak US dollar, US trade imbalance problems, and the US national debt continue to impact world economies and stock markets. While a good deal of the problem can be attributed to the trade imbalance and the national debt, the whole economic picture is really much more complex and global.

The trading nations of the world need to maintain overall free trade to stimulate employment and healthy economies, but reasonable steps should be taken to encourage more exports from those countries with large trade deficits, and more imports to those with large trade surpluses. Tariffs, trade embargoes, and import restrictions will lead to the trade wars seen earlier this

The mailbag

April (TRON Special Issue):

"All the articles contained in this issue are very interesting. I would like to continue reading *Micro*'s issues with a high level of articles." M.F., Argentina

June:

"I liked AMORE, evaluating microprocessors." V.M.J., Bombay

"I liked Capability-based System, Futurebus (IEEE 896) articles. I would like to see (articles on) Futurebus, cache coherency protocol, cellular processing, object-oriented systems." S.G., Bangalore, India

August:

"I would like to see applications of Modula-2 on small systems. Product reviews of Modula-2 compilers." A.R., Buenos Aires

"I liked the 80387 and its applications and the guest editor's introduction article. I would like to see more microprocessor interfacing." A.O.F., Tucuman, Argentina

"I would like to see biomedical applications in *Micro*." A.P., La Plata, Argentina

"I would like to see a survey of projects/applications of the Inmos transputer T414/T800." B.M., Neubiberg, West Germany (Inmos has greatly reduced its operations in Colorado in the US. I will ask the European *IEEE Micro* editorial board members to follow up on future Inmos projects.)

"I liked the feature, 'Introduction to the Clipper Architecture' and special feature, 'VLSI and System Performance Modeling.'" J.M.K., Bangalore, India

"Dear 'Lonely' Jim, At last a review of the Intel 80387, but still you like Motorola better, don't you (68020's, 68881 you put on the front cover!)? I liked Am29000 applications, please—great article. I would like to see names of 32-bit micros on front cover—be a bit more controversial...(you'll get more responses). (a) IBM PS/2 50/60/80 new bus, (b) more industrial applications—real engineering one-chip micros. (Enough mainframe micros for now!) P.S. TRON project not so 'big' here in Japan. Japanese software—don't touch, please!" J.T., Tokyo (Your assertions are not correct. First, all of the articles for the August issue were solicited by the guest editor, not me. They were all reviewed by disinterested peers in accordance with Computer Society policy. Since I work in the semiconductor industry, I never participate in decisions when a product is displayed on *IEEE Micro*'s cover. Personally, I feel the very best covers are the cartoons.—J.F.)

century. Worldwide depression resulted.

A lot of really exciting technology is being developed worldwide right now. We all hope the world economy will remain healthy, and research and development will continue. *IEEE Micro* plans to bring you articles on these topics in our usual in-depth style throughout the year.

Dick Stern, our extremely well read MicroLaw editor, is recovering from back surgery. I am sure that you will join the *IEEE Micro* editorial board, staff, and me in wishing Dick a speedy recovery.

Again, we have quite a few comments in the mailbag. We certainly appreciate your comments. I always try to publish all comments received, with as little editing as possible. I edit lengthy comments when I feel that I do not change the message substantively, or when I feel a message would be offensive to one group. I never edit comments critical of the magazine, the editorial board, or me.

Marie English, your managing editor, has been kind enough to sort the comments by chronological order, oldest first. It seems the worldwide mailing systems are pretty slow.

Best regards,



Jim Farrell

"I liked MicroLaw, New Products." R.M., Warszawa, Poland

"I liked the great article on hardware syntax processing, and the Argies [sic] on page 3! Also, the feature articles and the cover. I disliked nothing. A great issue! Keep it up! I would like to see more on symbolic processing applications of standard micros. (I'm a bit overspecifying, aren't I?) and coverage of industrial applications of new machinery (32032, 68020)." A.P., Buenos Aires

"I liked system consideration in the design of the Am29000." B.K., Albany, CA

"I liked 32-bit microprocessors articles." M.J., Rio de Janeiro

"I liked Marlin H. Mickle about the PC neurocomputer.... I would like to see further papers about neurocomputers." K.V., Bratislava, Czechoslovakia

"I liked the text of VLSI hardware." S.S., Istanbul

"I liked Clipper—one of the best presentations of microprocessor architecture I have ever read." J.P., Warszawa, Poland

"I would like to see a comparison of the VMEbus, Futurebus, Nubus, and Multibus II compared using the same cards; i.e., How does one bus rate against the others in the same environment." J.W.C., Denton, TX (I have barely recovered from our last engagement in the "Bus Wars"; however, I will try to find a comparative article.—J.F.)

"I would like to see more articles on Forth and Forth machines (Novix, etc.)." T.M., Milano

"I liked the clear and appealing presentation of 'Introduction to the Clipper Architecture.'" N.F., Zurich (I am noting the continuing affirmative comments on this article.—J.F.)

"I liked 'VLSI and System Performance Modeling.' I would like to see more CAD about VLSI." M.F., Bandung, Indonesia

October:

"I liked 'From the Editor-in-Chief.'" J.S., Bedford, Nova Scotia

"I liked the 'IMS T800 Transputer.' Clear and very readable magazine. Keep up the good work. I disliked two thirds of 'From the Editor-in-Chief'—was fill. 'European Cooperation in Information Technology Industry': biased towards UK. I would like to see a comparison of high-level languages for parallel processing and a review of image processing hardware/processors." S.G., Newcastle, UK (I suspect the managing editor placed these two comments in this order so that I would not suffer from an inflated ego for an unduly long period of time. Actually, I sincerely appreciate both.—J.F.)

"I liked 'A Low-Cost Distributed Architecture for Telecommunication Systems.' I would like to see about telecommunications and data communications." B.G.Y., Inchon, Korea

"I would like to see computers for process control." J.M., Rio de Janeiro

"I liked DOOM and T800 articles." R.P.S., Siegen, West Germany

"I have enjoyed Richard Stern's MicroLaw for years. Always read it first!!! Truly excellent, outstanding." R.M.R., Chautauqua, NY

"MicroLaw was outstanding as usual: you'll make an engineer out of Stern yet; transputer article was good; New Products and MicroNews were fine—and useful. I disliked DOOM paper...indecipherable...as was the fault-tolerant architectures paper (pg. 47 was not explained adequately). I would like to see articles on OS/2; articles on reverse engineering or simply understanding the ASIC chips in IBM's PS/2 line—what they do and how, timings, etc.; worldwide electronic mail nets and impact." T.T., San Pedro, CA (For the record, in addition to a law degree from Yale, Richard Stern also holds a BSEE from Columbia. I am concerned about your comments about the DOOM and fault-tolerance articles. Many readers liked both, but others—not directly involved in the disciplines, I suspect—found them difficult. Perhaps we should ask authors to provide a brief tutorial on important topics that do not have a wide following. I would appreciate your further comments.—J.F.)

MicroView

CISC, RISC, WISC: What's in a name?

Christine Miller, Assistant Editor

Not that we feverishly collect acronyms at *IEEE Micro*, but WISC is a new one on us. CISCs and RISCs have made their marks; WISCs looked like something to investigate.

We spoke with George Smith, president, and Edward Bryan, vice president of software development, International Meta Systems, about their writable instruction set computer. The Max 2 functions as an attached processor for IBM PC ATs, or other machines. According to the company, the low-cost WISCs compete with mainframes and superminis in scientific, engineering, and high-level programming applications.

First of all, can you briefly define writable instruction set computing for our readers?

We are able to change the fundamental nature of the way a computer operates by changing the microcode in its instruction memory. The WISC's unique architecture contains three memories: context, microcode instruction, and user. Each memory has a separate bus which provides a very high bandwidth into the CPU. The aggregate bandwidth is 180M bytes/s.

Users can define their own applications to embed in the firmware. We first design the machine for, say, Fortran, C, Prolog, or Lisp. We optimize the engine so it can run faster because we've tailored it to a software program. Its memory is large enough to let users write their own code. Users can then add to the instruction set of the machine and create what we call an application-specific instruction set, or ASIS.

"Users can define their own applications to embed in the firmware."

The board itself plugs into an IBM PC AT or other host as an attached processor computer, or it can function as a complete computer program except for I/O. The board can be adapted to Unix or any other kind of host by redoing the I/O and electrical interface. We chose the host method because we're a small company and developing an operating system can run into the $15-20 million range.

What implications do WISCs have for the microprocessor industry?

A whole new class of machines is being developed. WISCs open up an era of a new kind of computing, while taking full advantage of RISC technology. RISCs have been a limited success because you can't keep the machines going at full speed. With three buses and three memories, that technology can reach its full potential.

What is the relationship between a WISC and a RISC?

RISC refers to the complexity of the CPU element. Our CPU (a proprietary chip) is a RISC-class machine with 24 distinct classes of instructions. The CPU is a basic building block for writable instructions. The advantage of a very simple machine is that you can make it run very fast. The board operates at 20-40 MIPS. When you put the WISC on top of that, it takes 5 to 10 microinstructions to execute one very high-level instruction. In a complex instruction set computer (CISC) it takes 20 to 30 microinstructions to perform one high-level-language instruction.

What are the advantages of WISC over other instruction-set processes?

Most machines are designed for universal applications. A WISC set allows the user to be both universal and application specific. That gives the user an advantage. Users can now optimize their machine for one application and change the instruction set to another application in 80 ms through dynamic loading into the static RAM.

What does reprogramming a WISC for new applications involve?

It's like writing subroutines in assembly language. First, subroutines are selected to be written in an HLL which will be compiled, and then those subroutines are translated into microcode and loaded into instruction memory.

How long does it take to learn to do this?

Two weeks to a month.

All in all, what would motivate a user to choose the WISC approach?

The potential for much, much higher performance at minimum cost. At $9000 per MFLOPS (Linpack), WISC is the approach of price performance for computationally intensive programs. For example, Apollo is $30-70,000 per MFLOPS, VAX is $90-100,000 per MFLOPS, while mainframes range between $300,000 to $400,000 per MFLOPS. The IBM host is a commodity machine right now, and the WISC provides a mainframe level of performance in a microprocessor.

Also, WISC performance enables fifth-generation languages for high-productivity software development. Smalltalk, which is a wonderful human-interface language, is an example.

"At $9000 per MFLOPS, a WISC board competes with a VAX costing $90-100,000 per MFLOPS."

What applications does a WISC support?

There are two levels: HLL applications for which we are providing internal instruction sets for fourth- and fifth-generation languages, and specific applications where the machine is particularly powerful. Scientific and engineering applications include finite element analysis, circuit simulation, simulation of physical or logical systems, and adapting Fortran to perform calculus. The WISC supports the solution of optimization problems anywhere, including the fields of chemical analysis, manufacturing, and economics.

We have placed the Max 2 with Schlumberger, TRW, Northrop, Zenith, NCR, The Aerospace Corporation, Dupont, Stanford University, UC Berkeley, and the Swedish Telephone Company.

Where did the idea of the WISC originate?

With Arthur Speckhard, formerly of the The Aerospace Corporation, now at IMS, who developed the architecture to support fifth-generation HLLs.

What led you to pursue its development?

In 1980, the IMS group was at The Aerospace Corporation, a federal think tank created to serve the Air Force space program. The Aerospace Corporation was trying to reduce the cost of software, yet had to have very high level languages. In order to get the machine to adapt to the very HLLs, we had to have writable instructions. Fifth-generation languages like Smalltalk, Lisp, and Prolog can't be compiled out. Traditional approaches don't work for them. The writable instruction set changes instructions which implement the bit-code instructions, or P-code language, in which they are written. The WISC recognizes and interprets P code.

In 1985, we obtained a release from The Aerospace Corporation and agreed to pay equivalent royalties to them and the US government. After forming IMS with private funding, we introduced the Max 2 WISC in 1987.

Is anyone else implementing WISCs?

Not that we know of, now. Some time back there was some development, but they didn't have a machine like ours.

What are the disadvantages of WISCs? What are you working to improve?

Our challenges? We want to move to higher clock speeds and higher memory speeds, and therefore higher instruction rates. These advancements are solvable in a straightforward way due to the non-von Neumann architecture of Max 2. Also, a different architecture demands the design of new software. Until you develop that software, you're at a definite disadvantage in the market. We have completed the Fortran compiler; the compilers for C, Smalltalk, Prolog, Lisp, and Pascal are under development. Eventually, we plan to develop an APL compiler.

The Max 2 WISC can execute two microinstructions per clock cycle. What does that allow?

It has a four-stage pipeline, which allows a higher number of instructions. You can do a test and branch, or a logical operation and a test and branch, in a single instruction. We don't know anything else that does it in this way.

You mentioned that the Max 2 has a non-von Neumann architecture. What do you call it?

Well, it goes beyond a Harvard architecture—no name has been coined. We could call it the Speckhard architecture—he might like that.

Does a WISC interface with other developing technologies?

The architecture of the machine dovetails nicely with advancing silicon foundry technology, which now produces 2.0-micrometer CMOS chips. We're going to adapt the Max 2 in six months to 1.5 micrometers. This will provide a 35-MHz clock rate. The submicron ranges are coming along, which will affect CPU performance, and instruction and context memories. We will be able to speed up to the 100-MHz clock kind of range.

But basically, the WISC does not depend on technological heroics for its performance. Although superconductive materials could be helpful, there is no real need for them.

"The advantage of a very simple machine is that you can make it run very fast."

On the other hand, neural networks utilizing one hundred—or thousands—of computers working together would be looking for small CPUs. We don't know if anyone in that field is even considering that, but it would be interesting if each neuron had its own instruction set.

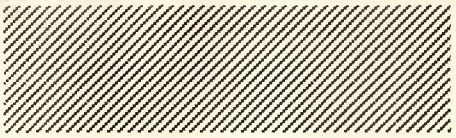
Are there any other developments on the horizon?

Implementations of direct execution in calculus, exact derivative calculation, and interval analysis. These will be available in late 1988.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 186 Medium 187 High 188

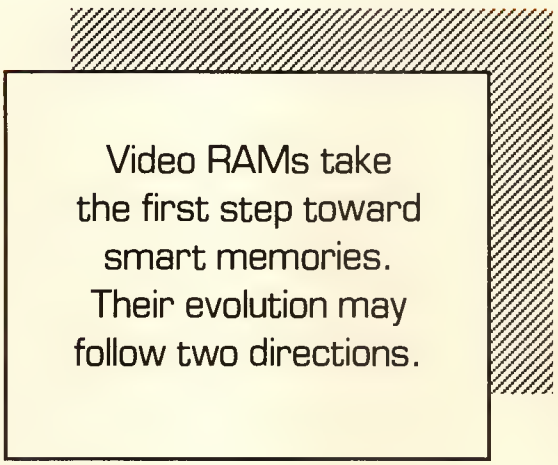


F E A T U R E

Video RAMs:

S T R U C T U R E A N D A P P L I C A T I O N S

Jean-Daniel Nicoud
Ecole Polytechnique Federale de Lausanne



Video RAMs take the first step toward smart memories. Their evolution may follow two directions.

A processor usually accesses memories through a single port of communication. When the processor passes information to a peripheral, it either performs the transfer itself or suspends its activity while a DMA device transfers the information.

Dual-port memories are more efficient, since both the processor and the device can access them "simultaneously." However, some limitations exist. It is not possible to write on one side and read the same data on the other at a given instant of time (100-200 nanoseconds). True dual-port memories also require the duplication of the address and data bus, and are expensive.

Screen displays that must be refreshed from information taken at a rather high rate from memory often use dual-port memories. The principle is well known: data is read and loaded into a shift register, which more or less directly outputs the video signal.

On personal computers like the Macintosh, the screen image is taken from main memory using DMA cycles. This is not the best solution, since these cycles remove up to 35 percent of the processing power when they load the display shift register. Wider memory words (when viewed from the display interface) lower the frequency of access or increase the bandwidth. They also increase the number of components significantly.

Memory manufacturers proposed the first solution in the early 1980's: accelerate the access by performing burst transfers with the memory (nibble mode, ripple mode, and static column, as we discuss in the section on dynamic memories). A much better solution, which avoids most of the wiring associated with shift registers, is to use video random access memories (VRAMs), which provide fast, block-transfer access to the internal memory.

VRAMs, or "dual-port dynamic RAMs," are dynamic memories that include a shift register for providing a high-speed secondary channel. In 1981, Texas Instruments initially proposed VRAMs with a 16K × 1-bit-wide organization. Mitsubishi, Hitachi, TI, NEC, and Fujitsu now offer them with a 64K × 4 organization and an impressive set of internal functions. Figure 1 lists VRAM signals. The standard package is 24-pin, dual-in-line, with the new width of 0.4 inches. A few manufacturers document a compact, "zigzag-in-line," 10-mm-high package, which is not yet easily available. Especially in this case one should be cautious with the large heat dissipation of VRAMs when they are closely packed together. Single-in-line, hybrid modules increase the functionality for the pin count, as later discussed.

0272-1732/88/0200-0008$01.00 © 1988 IEEE

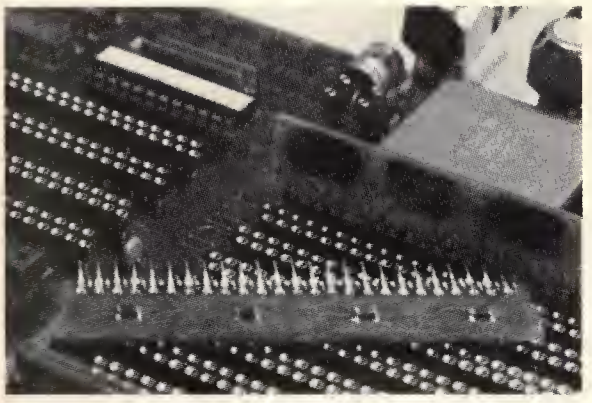
As shown in Figure 1, the first group of VRAM signals correspond to the usual signals of a 64K × 4 dynamic RAM (DRAM):

- Row Address Strobe ($\overline{RAS}$),
- Column Address Strobe ($\overline{CAS}$),
- Data Transfer ($\overline{DT}$),
- Write ($\overline{WR}$),
- Eight multiplexed address lines (Ai), A7...A0, and
- Four bidirectional data lines (Di), D3...D0.

The serial access memory (SAM), which consists of four 256-bit registers (Figure 1a), requires six additional lines. A serial clock (SC) and a Serial Output Enable ($\overline{SE}$) signal control the shift register's I/O lines, S3...S0. The $\overline{DT}$ (or the $\overline{DT}$ / $\overline{OE}$) signal is used in conjunction with the $\overline{RAS}$, $\overline{CAS}$, and $\overline{WR}$ signals to transfer data between one selected memory row of 256 locations and the shift register (detailed operations follow later). Hence, VRAMs are not real dual-port memories since a cycle must be triggered on one port to allow a transfer on the other port. However, they do present an adequate compromise which fits especially well with graphics display applications and shows promise for applications in other interesting fields.

VRAMs also support a mask write feature, depending on the timing of the $\overline{WR}$ signal (also named $\overline{WB}$ /$\overline{WE}$).

Compared with a simple DRAM of the same capacity, a VRAM has a bigger package: both longer (due to six more pins) and larger (due to a big die size). The power supply is significantly larger, but the price per package is about the same. Taking into consideration that some simple DRAMs have four times the capacity of VRAMs, a VRAM board is about four times as expensive and eight times larger and more power-hungry than a simple DRAM board of the same memory capacity. However, displays do not need large memories and the comparison with DRAMs—including all the logic and shift registers—favors VRAMs, especially for high-resolution screens.



Hybrid VRAM modules (Valtronic) on color graphics board.

VRAMs are found in a growing number of applications besides displays. We have developed several projects at Lausanne to test VRAMs and to define the best architecture for hybrid modules. Here, we first briefly describe three typical VRAM applications, then discuss simple dynamic memories and the features of VRAMs, and finish with a further description of applications.

VRAMs are not easy to implement, but they belong to the group of components design engineers must know and be ready to use when applicable.

Graphics displays

With their serial ports, VRAMs are well suited for black-and-white or color graphics displays. We do not consider alphanumeric displays here; they belong to the past. VRAMs have too large a capacity for screens of only 2000 characters. For all graphics display schemes, several VRAMs can be placed in serial and parallel to reach the required bandwidth

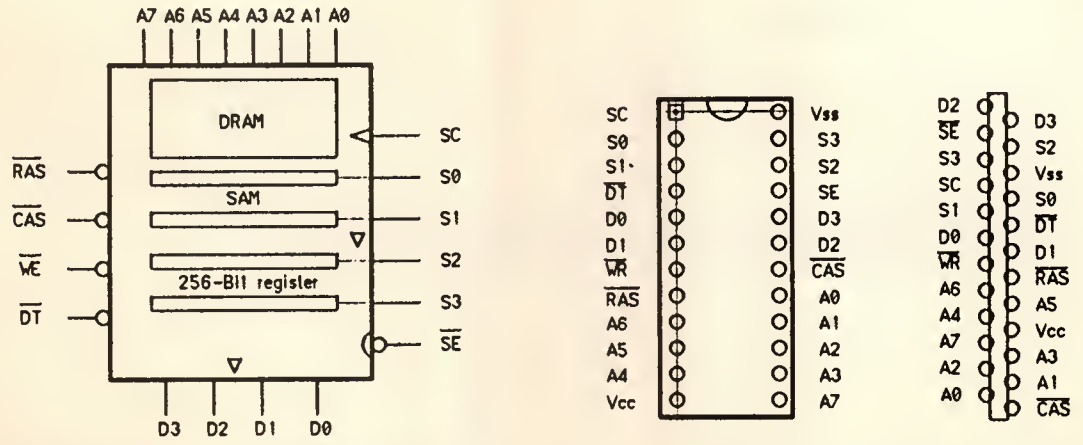


Figure 1. VRAM block diagram and typical pinout.

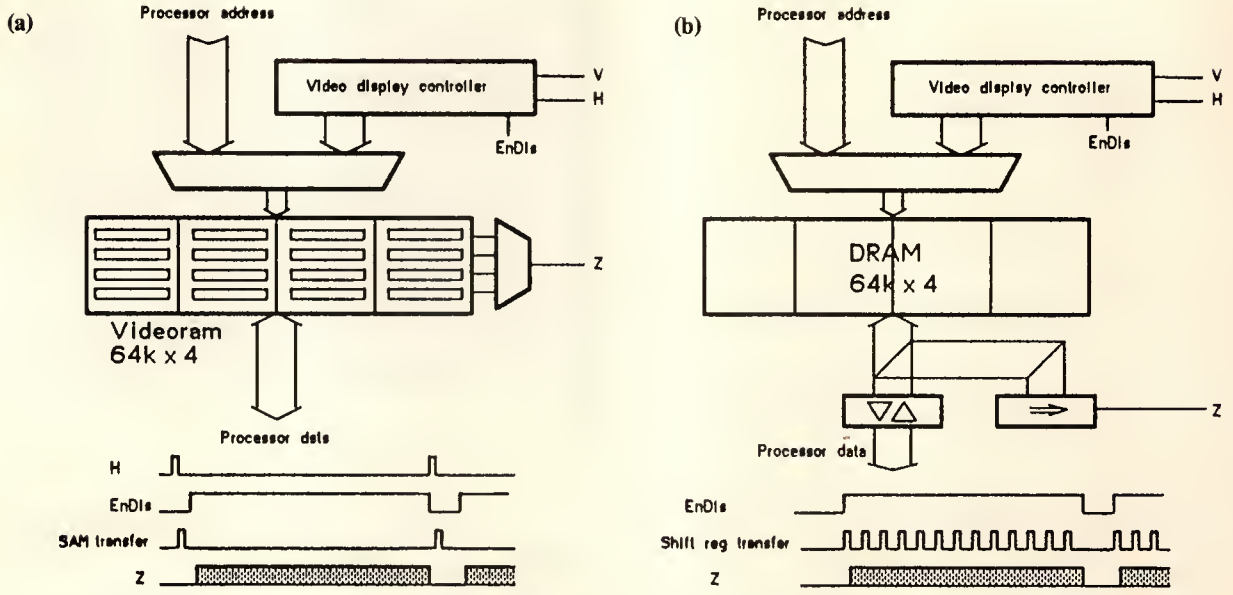


Figure 2. Bitmapped graphics display with black-and-white screen interfaced to a 16-bit bus (a) and traditional display interface (b). V is the vertical retrace pulse, H is the horizontal retrace pulse, EnDis is the enable display line ($\overline{\text{Blank}}$), and Z is the intensity.

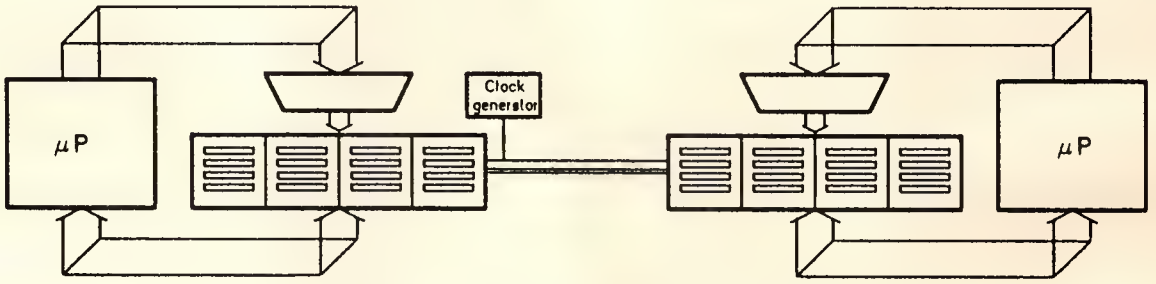


Figure 3. Fast block transfer between computer memories.

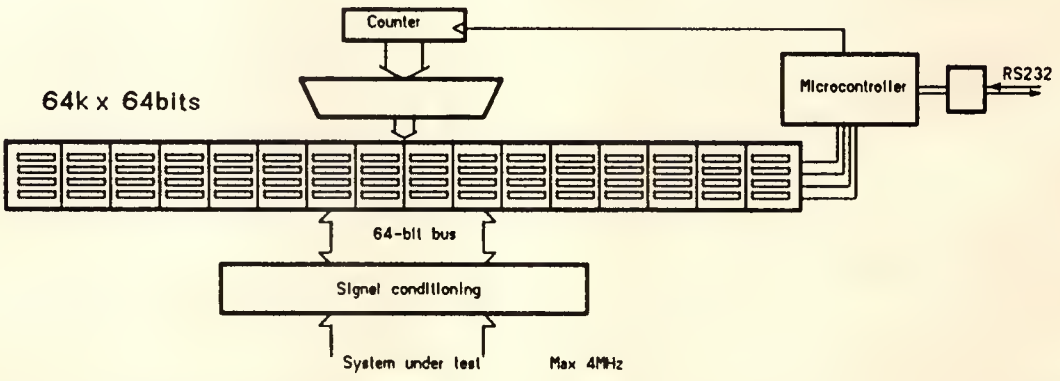


Figure 4. Test vector generator and logic analyzer.

and bitmap size. Figure 2a shows a black-and-white screen with good resolution (70 MHz, 1024 × 800 dots) interfaced to a 16-bit bus. It requires only four 64K × 4-bit VRAM circuits.

Compared with a traditional display interface (Figure 2b) in which the data from memory is transferred 16 bits at a time into a transistor-transistor logic (TTL) shift register, the hardware savings corresponds to a few shift registers and drivers—or about one third of the area. A far greater savings is obtained on bus bandwidth requirements. The 70-MHz screen on a 16-bit bus saturates the bandwidth of normal, dynamic memories. A VRAM interface requires only a small percentage of the bus bandwidth, and the synchronization with the processor is easier since the transfer occurs during the horizontal retrace.

Most available video display controllers present a problem: They are too old to incorporate VRAM support.[1] Their architecture is optimized for multiple on-line fetching of data, as in Figure 2b. No signal exists to fetch the content of the first displayed line in advance. Also, the address does not correspond to the beginning of the next line during retrace.

Using VRAMs also simplifies color displays, as we discuss in the section on color-screen organization. The TV camera and scanner interface can benefit for similar reasons as the displays, but needs VRAMs with shift-in capabilities. All but one of the vendors have this feature.

Fast link

The dual-port nature of VRAMs and the fast transfer speed offered by their shift mechanisms encourage their use in the design of communication interfaces. Figure 3 shows a typical block diagram. We discuss it in more detail in the section on multiprocessor communications.

Logic analyzer

The communication link is not necessarily the serial port. One can imagine applications where a simple microcontroller fills or empties the memory by shifting the information in and out. The main port is used under hardware control to obtain some information transfers on a wide bus, such as for a tester or logic analyzer (Figure 4). The advantage of the parallel port in this case is that it allows random-access and individually controllable read and write cycles by groups of 4 bits.

Dynamic memories

The principle of dynamic memories is well known,[2,3] and we only briefly repeat it here to improve the understanding of VRAMs. Figure 5a provides the DRAM model. For 64K DRAMs, memory cells are placed on a 256 × 256 grid; they

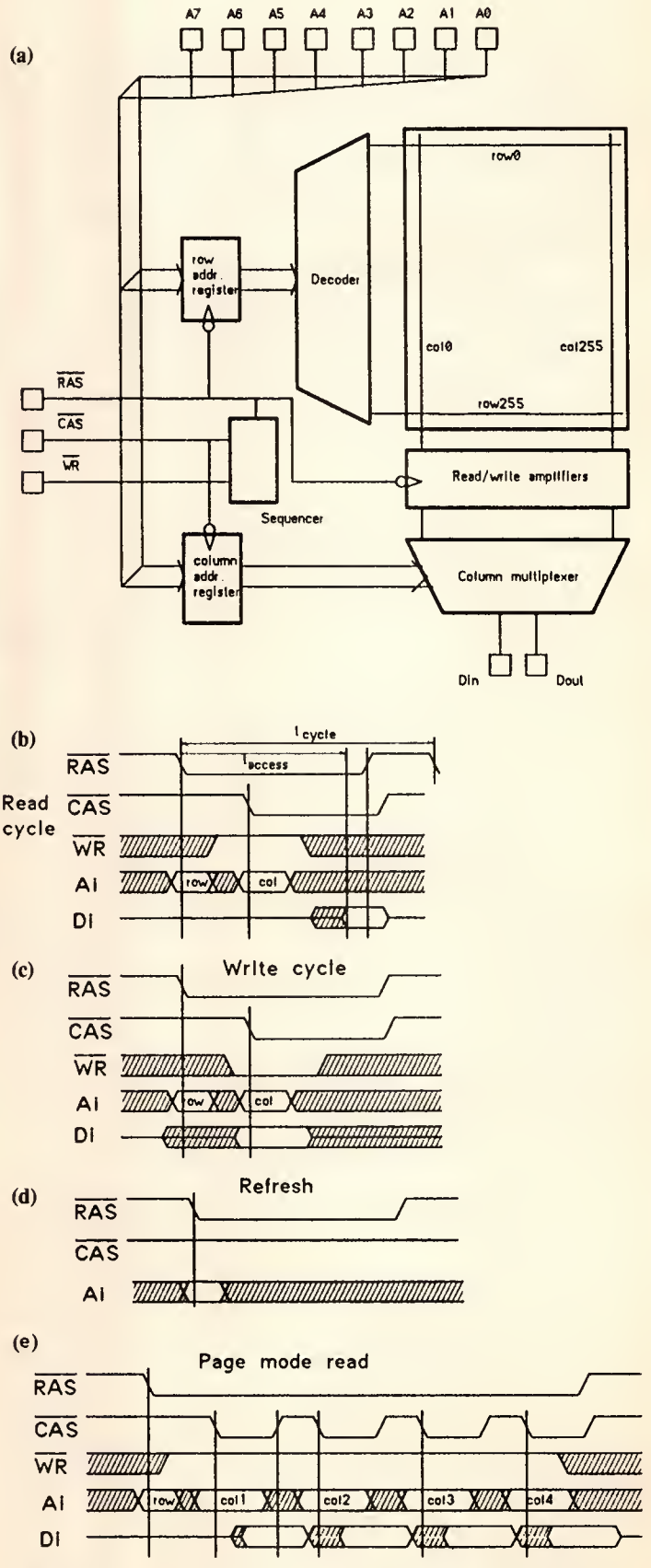


Figure 5. DRAM model (a) and major modes (b) through (e).

are selected and refreshed one row at a time. The 8-bit row and column addresses are multiplexed without performance loss. The access time corresponds very well to that of most microprocessor applications. The only problem is a rather long recovery time (idle time between two accesses). Typically, the cycle time is almost double the access time (Figure 5b).

A read-modify-write cycle is possible and is explained for VRAMs in a section on the read-modify-write mode. To improve the throughput, page-mode access (Figure 5e) accelerates the transfer of consecutive bits of the same row. When the internal architecture provides four arrays of memory cells with separate amplifiers, a faster multiplexing of the internal outputs is possible. This is named nibble mode[4] or ripple mode.[5] The timing diagram is very similar to page mode—but with a higher frequency of the $\overline{CAS}$ signal—and is limited to four cycles in the nibble mode.

Most recent dynamic memories offer static column addressing. It is also a variant of page mode, but the $\overline{CAS}$ does not pulse: a random change in the column address immediately brings the corresponding data. All these solutions have lost their past interest for display applications.[6] Transfers may be accomplished as fast as with VRAMs, but no other access can be made during transfers.

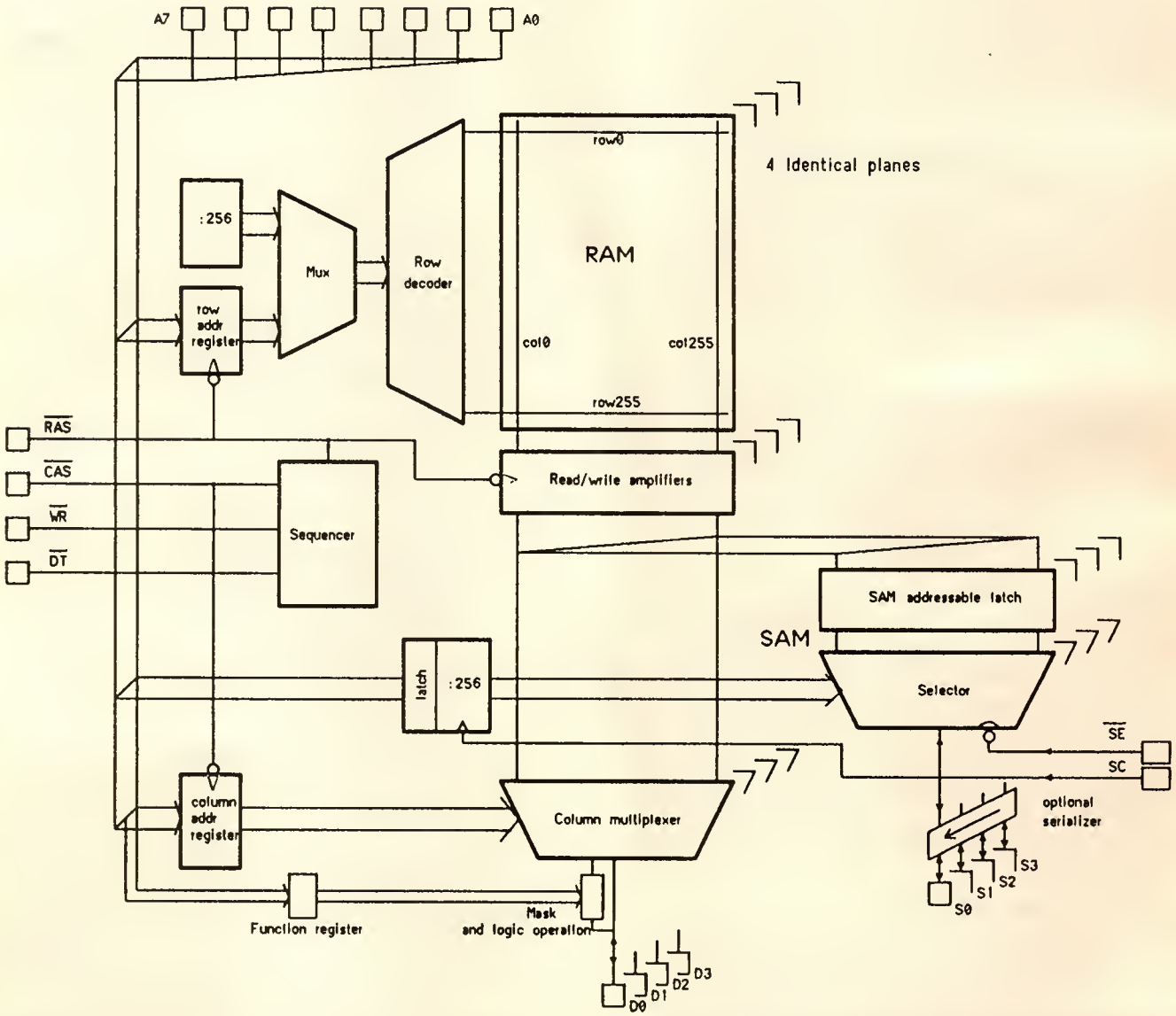


Figure 6. VRAM block diagram.

Table 1. Signal names and features of available VRAMs (end-1987).

Signals and features	Mitsubishi M5M4C264	Hitachi 53461	Hitachi 53462	Texas Inst. TMS4461	NEC μPD41264	Fujitsu MB81461
$\overline{WR}$	$\overline{WB}/\overline{WE}$	$\overline{WE}$		$\overline{WE}$	$\overline{WB}/\overline{WE}$	$\overline{ME}/\overline{WE}$
$\overline{DT}$	$\overline{DT}/\overline{OE}$	$\overline{DT}/\overline{OE}$		$\overline{TRG}$	$\overline{DT}/\overline{OE}$	$\overline{TR}/\overline{OE}$
$\overline{SE}$	$\overline{SE}$	$\overline{SOE}$		SG	$\overline{SOE}$	$\overline{SE}$
SC	SC	SC		SC	SC	SAS
Di	Wi/IOi	IOi		DQi	Wi/IOi	MDi/DQi
Si	SIOi	SIOi		SDQi	SOi	SDi
Mask write	Yes	Yes		Yes	Yes	Yes
Shift input	Yes	Yes		Yes	No	Yes
Logic operation	No	No	Yes	No	No	No
1024 × 1 mode	No	No	Yes	No	No	No

VRAMs

VRAMs are still difficult to use because their specifications are not always complete. Minor differences between manufacturers impede sophisticated applications. Hitachi, however, is a clear leader in the field and provides good specifications and application notes.[7]

General structure. In addition to the DRAM, VRAMs include a SAM, which consists of four serial access registers named SAM registers (Figure 6). The SAM looks like a shift register and allows one to shift the register's data in and out. Special, internal data-transfer cycles transfer a memory row with the SAM register.[8]

The address of the first bit of the SAM register to be transferred can be specified, since the SAM register is not a usual shift register but rather a parallel latch with addressable read and write logic. A presettable counter points to the transferred bit; this counter scans the latch in a circular manner.

An additional control signal, $\overline{DT}$, is required for the transfers between the DRAM and the SAM. Depending upon whether $\overline{DT}$ is activated after or before $\overline{RAS}$, a normal memory access or a SAM data transfer occurs.

One more function is provided in normal write cycles. A temporary Mask Register is loaded at the $\overline{RAS}$ active edge if the $\overline{WR}$ signal is already active. The decision in favor of a read or a write cycle is made on the $\overline{CAS}$ active edge, as in all dynamic memories. This masking function is highly useful for color displays.

Hitachi offers a more sophisticated function for its VRAM parts to allow an internal logic operation between the memory-cell content and the written data. This function helps the bit-blt operation[9] and avoids a longer read-modify-write cycle as well as the need for external logic, either dedicated or included in video controller chips.

Several differences exist between the VRAMs of the various manufacturers. NEC, the first to provide VRAMs, does not offer the capability of shifting in data. At the beginning of 1988, only Hitachi performed logic operations. It will take some time before the functionality of VRAMs is settled and before video controllers are designed to take full advantage of that functionality. We hope that the future 256K × 4 VRAMs will be more standardized. Just previous to *IEEE Micro* press time, Hitachi announced the 256K × 4 VRAM HM534251, which does not have the logic function.

The signal notations differ slightly from one vendor to another as well. Table 1 shows the differences by vendor. $\overline{RAS}$, $\overline{CAS}$, and Ai are identical for all vendors. The composed names correspond to the meaning of the signal at both the $\overline{RAS}$ and the $\overline{CAS}$ negative edges.

Read and write mode. A read cycle on a VRAM is identical to the one on a DRAM (Figure 7). A write cycle is also identical if the $\overline{WR}$ signal is activated after the $\overline{RAS}$ active edge. We do not give the precise timings here since they are well documented by manufacturers and represent a considerable amount of information. The timing diagrams in all figures are approximately to scale. Typical access time is 120 ns and cycle time is 220 ns.

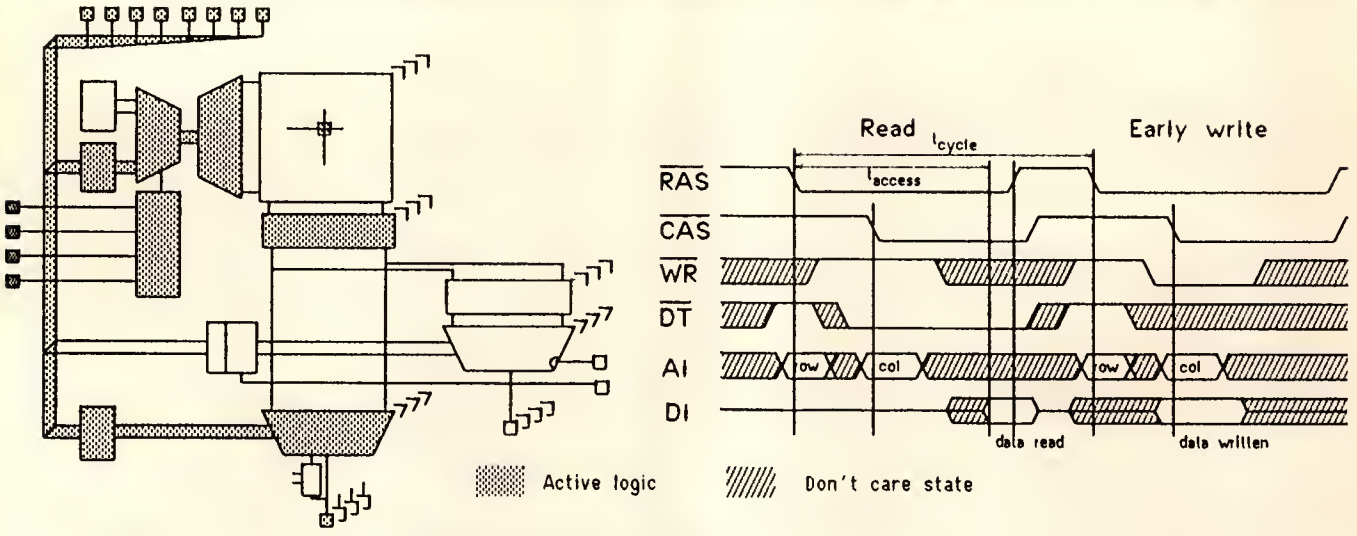


Figure 7. VRAM read and simple write mode.

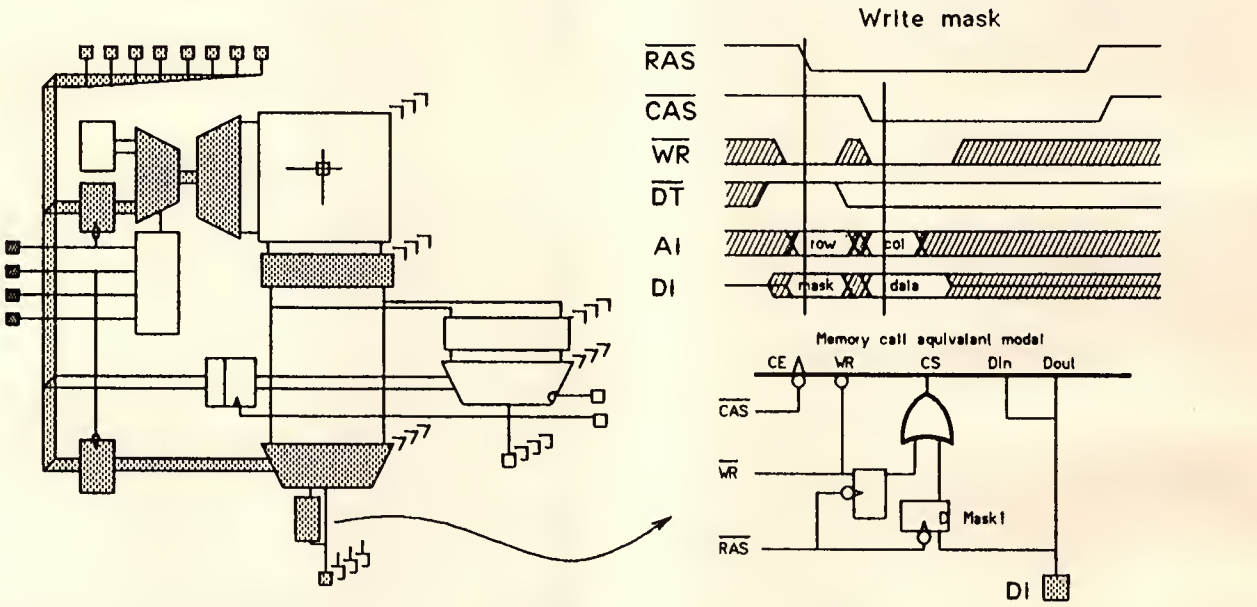


Figure 8. VRAM write mask mode.

Write mask mode. As previously mentioned, an interesting feature of VRAMs is the possibility of a masked data transfer requiring the multiplexing of both the mask and the data. The $\overline{RAS}$ active edge stores the temporary enable mask (Mask1), and the $\overline{CAS}$ active edge transfers the data, but only when the corresponding mask-enable bits are set at 1. See Figure 8 for an equivalent model of the write logic and the timing diagram.

For preparing and transferring the temporary mask on a microprocessor system, the TTL circuits 74xx646 or 74xx652 can be useful as a driver on the data bus since they include a transceiver and a register.

Logic operation mode. Hitachi's HM53462 proposes a very interesting logic operation possibility. We hope that all manufacturers will copy this operation, and it will be supported by powerful graphics display controllers.

During the refresh operation, a 4-bit, logic-function code is read on the address lines and an additional permanent enable mask (Mask2) is read from the data line (Figure 9). If the logic code is different from the "transparent" code (internally set at power up), a normal write of an information Di will be performed by the VRAM according to these steps:

- a data Mi is read from memory and remains in memory;
- a logic function is performed between the memory data Mi and written data Di; and
- the result is written in memory if the corresponding Mask2 bits are set.

The write mask mode also exists when $\overline{WR}$ is activated at the $\overline{RAS}$ active edge. It is compatible with the one previously explained. During write mask mode operations, logic functions are automatically inhibited and Mask1 is used instead of Mask2 (this does not appear in the equivalent model of Figure 9).

Table 2 lists the logic-operation codes. The proposed scheme is very powerful since each access offers a choice between two masks. Mask1 is not provided by present-day processors or graphics controllers and must be stored inside the VRAM display interface. Mask2 is stored inside the memory circuit after a logic-operation definition cycle and is used unless the logic operation is transparent.

One of the logic-operation codes is also an operation code for the serialization of the 4-bit shifted data on a single pin (see Figure 6 again). This fact may slightly simplify the design of low-cost applications.

At initialization, the HM 53462 must be put in the required state by a correct sequence. Experience shows that, by default, the state at power up corresponds to the transparent video mode (53461 mode).

Read-modify-write mode. All dynamic RAMs have a read-modify-write mode which allows one to start a read cycle and write the data at the falling edge of the $\overline{WR}$ signal. On 1-bit-wide memories, separate input

Table 2.
Hitachi HM53462 logic-operation codes.

A3...0	Equation F (Di external data, Mi memory cell)	Explanation
0000	0	Clear data
0001	$\overline{Di} \cdot Mi$	AND
0010	$Di \cdot \overline{Mi}$	AND2
0011	+	1024 × 1 org
0100	$Di \cdot \overline{Mi}$	AND3
0101	Di	Transparent mode
0110	$Di \oplus Mi$	Exclusive OR
0111	$Di + Mi$	OR
1000	$\overline{Di + Mi}$	NOR
1001	$\overline{Di \oplus Mi}$	Exclusive NOR
1010	$\overline{Di}$	Invert
1011	$\overline{Di} + Mi$	OR2
1100	$\overline{Mi}$	Inv2
1101	$Di + \overline{Mi}$	OR3
1110	$\overline{Di \cdot Mi}$	NAND
1111	1	Set

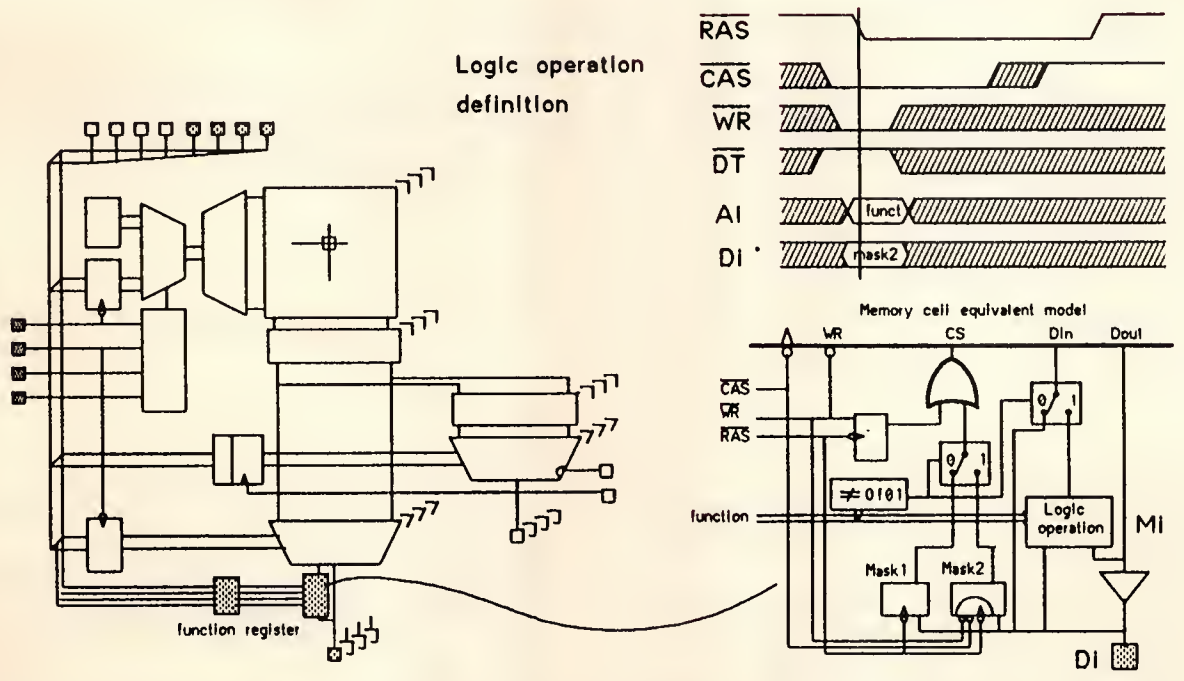


Figure 9. VRAM logic-operation definition cycle (Hitachi HM53462).

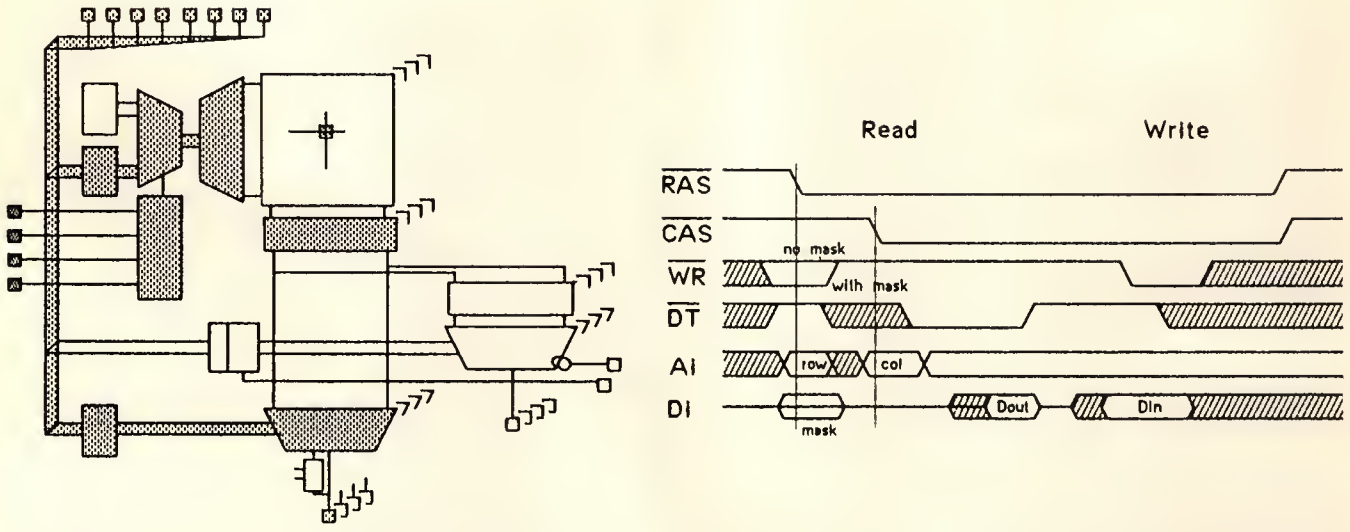


Figure 10. VRAM read-modify-write mode.

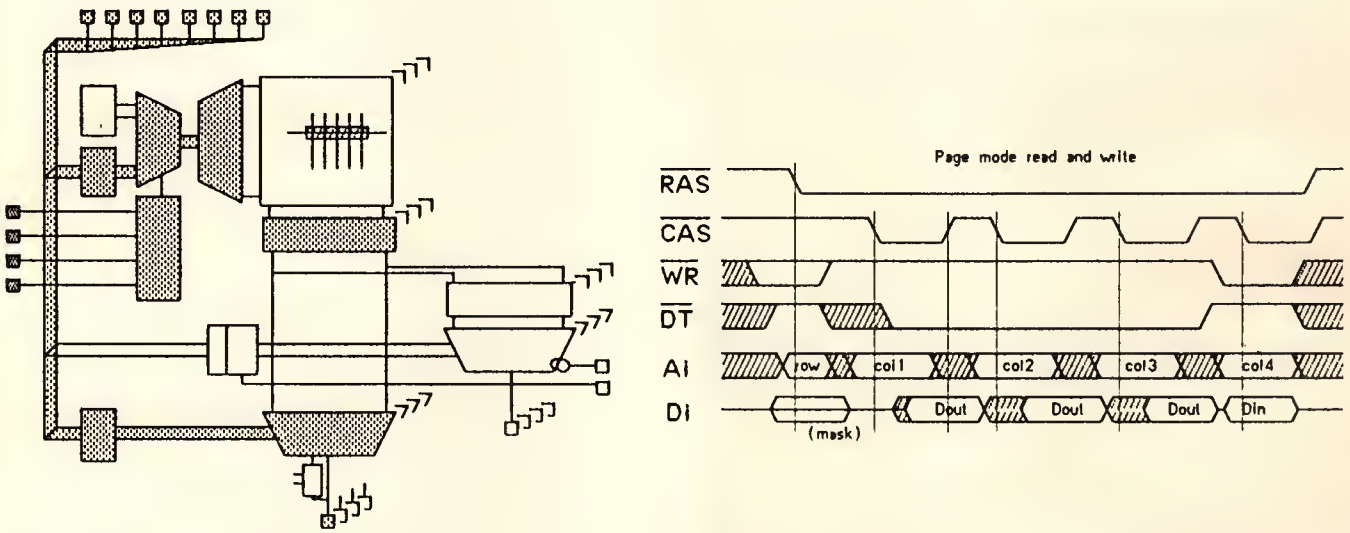


Figure 11. VRAM page mode.

and output data are used. On 4-bit-wide DRAMs and VRAMs, an Enable signal ($\overline{DT}$) disables the output before the preparation of data for the write cycle (Figure 10).

The read cycle does not have to be completed or activated before the write cycle starts. The delayed write cycles, or $\overline{DT}$-controlled write cycles, are documented separately by the manufacturers, but are not worth describing here in more detail.

Page read and write mode. Page mode allows one to read consecutive locations on the same row with some savings of time and power, compared to separate access. In the case of write cycles, as shown in Figure 11, an enable mask can be written at the $\overline{RAS}$ active edge and is applied to all the subsequent written data.

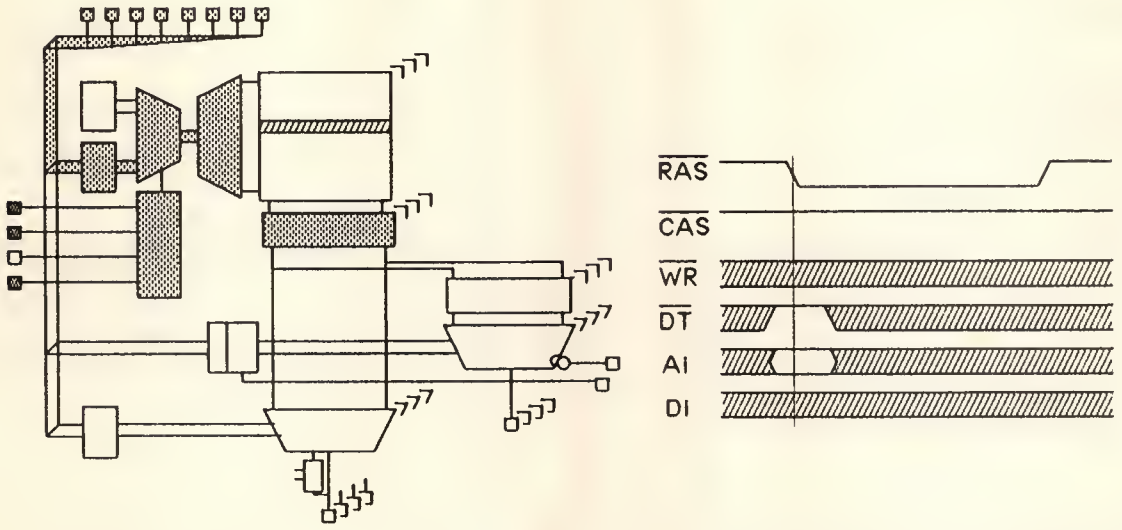


Figure 12. RAS-only refresh mode.

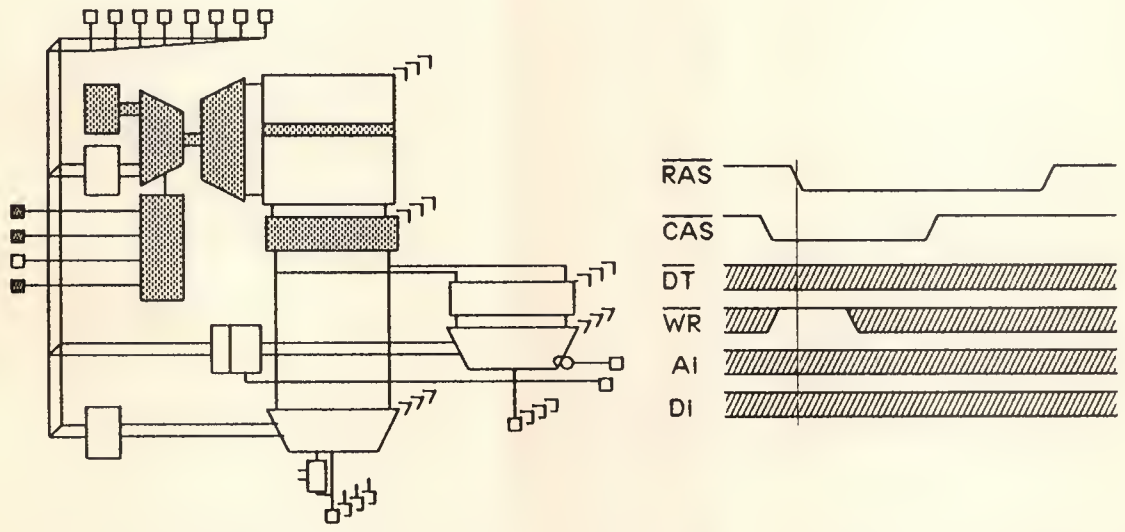


Figure 13. CAS-before-RAS refresh mode.

RAS-only refresh mode. All the cells of a dynamic memory need to be refreshed every 2 or 4 milliseconds. Each time a RAS negative edge occurs, the full row is refreshed. A solution offered on all DRAMs and VRAMs for refreshing the cells is to access consecutive rows once every 16 microseconds or in bursts, doing simple RAS-only cycles (Figure 12). This implies an external refresh counter and logic that are found inside most graphics controllers.

CAS-before-RAS refresh mode. On recent DRAMs and on all VRAMs, the refresh logic can be simplified thanks to an internal refresh counter. A special CAS-before-RAS cycle activates this counter (Figure 13). On parts having internal logic operations, one must be careful about the value on the WR line during refresh: WR must be high when RAS falls to prevent reloading of the function and Mask2 registers, as previously described.

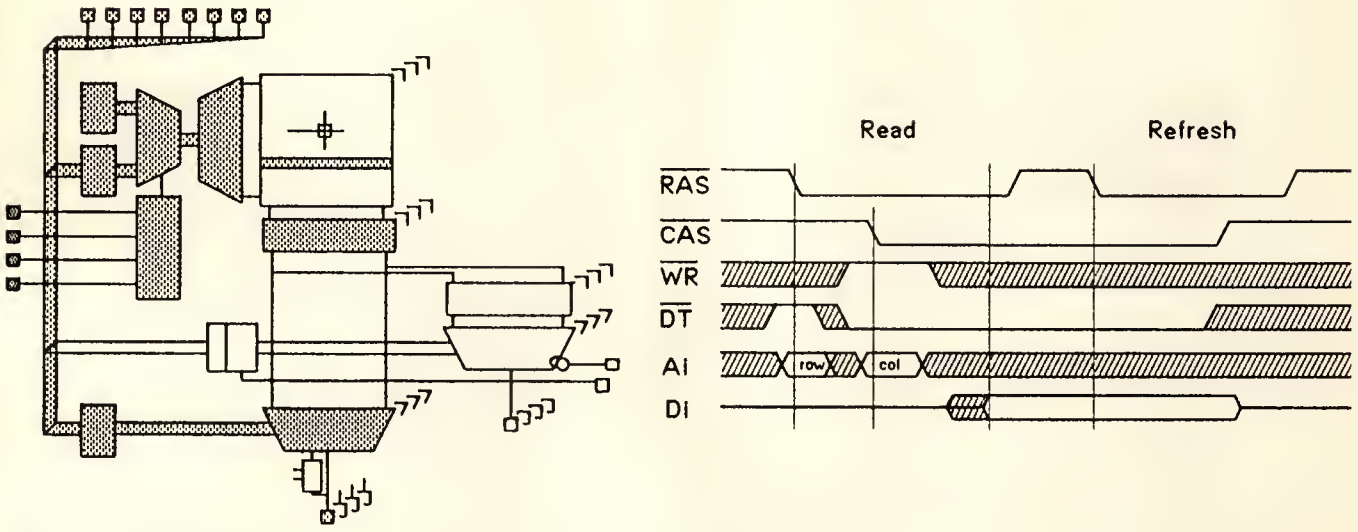


Figure 14. Hidden refresh mode.

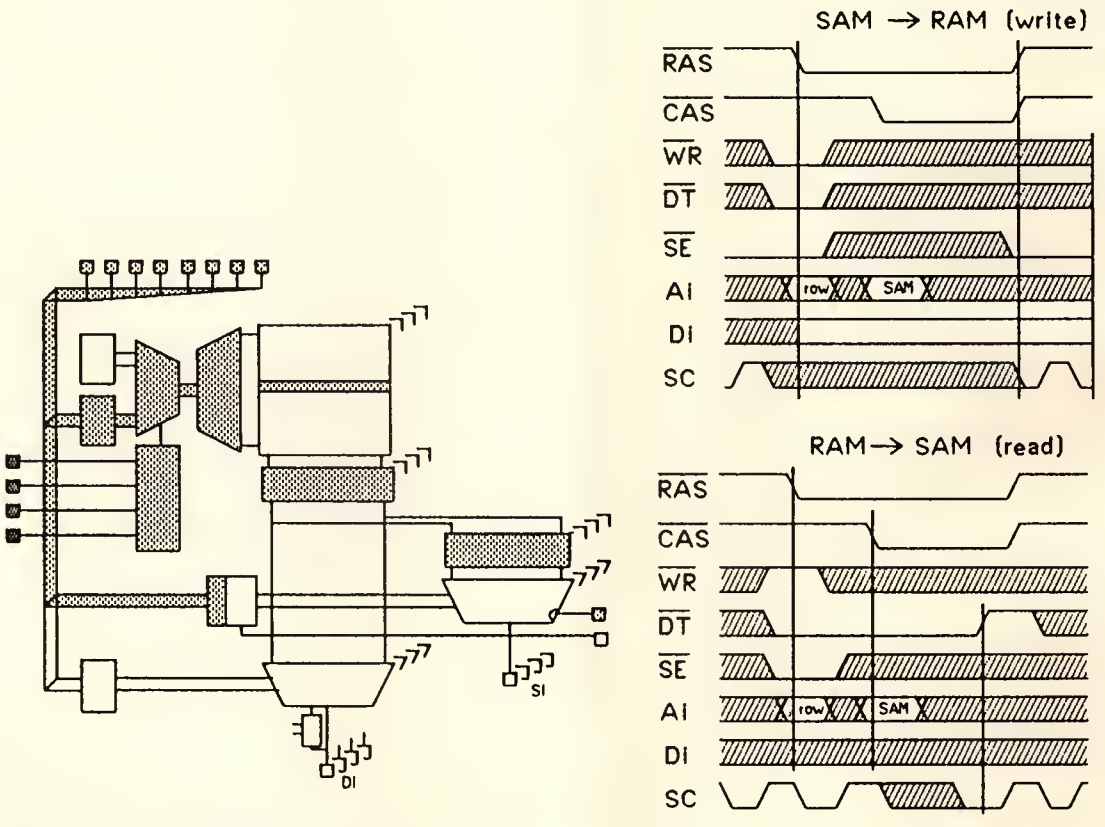


Figure 15. SAM read and write transfer.

Hidden refresh mode. On all DRAMs and VRAMs, the data is maintained on the outputs as long as $\overline{CAS}$ is active. A $\overline{RAS}$-only refresh cycle can therefore be started before the end of the cycle (Figure 14). The next cycle will not be able to start before the recovery time following the refresh cycle.

SAM transfer. The SAM register of a VRAM is loaded by reading a memory row with $\overline{DT}$ active at the beginning of the cycle (Figure 15). If $\overline{WR}$ is active at the $\overline{RAS}$ active edge, the SAM register content is written into the selected memory row.

The address at the $\overline{CAS}$ active edge is used to select

the start address of the serial transfer. In read mode, data will be shifted from that address. In write mode, the next data shifted in will be written into the SAM register from that address. Debugging SAM transfers is tedious due to the long set of operations necessary to check correct functioning. We recommend putting some test logic and programs into the first prototypes to help the debugging process.

These SAM transfers pose many problems. Procedures are poorly documented and partly incompatible between manufacturers. The internal SAM-RAM transfer occurs at different times in relation to the SC status, and one more clock edge is sometimes required.

SAM serial shift. The 256-bit, SAM register behaves like a simple shift register. The positive edge of the clock shifts the data out or stores incoming data (Figure 16). If the clock edge occurs when $\overline{SE}$ is not active, the counter is incremented, but the addressed SAM data is not read or written. It is also important to note that after a RAM-SAM transfer, the new data will be available on the serial output only after the first SC positive edge following the deactivation of $\overline{DT}$.

Several manufacturers also use a dynamic technology for the SAM cells. If the data contained in the SAM register is not to be altered, 256 clocks on the SC line must occur every 4 ms. In most applications this refresh is unnecessary because no delay occurs between the preparation of the SAM content and its use.

SAM on-the-fly transfer. The size of the SAM is sufficiently large in most video display applications to correspond to more than a complete displayed line. The transfer occurs during horizontal retrace, and there is no problem in alternating transfer cycles and shift cycles. A continuous stream of shifted information is possible with a very precise timing of the $\overline{RAS}$ and $\overline{DT}$ signals in relation to the signal SC. This may not be easy to implement with most microprocessor applications in which the transfer is defined by the processor or a DMA unit with considerable jitter.

In a read-transfer cycle, the row is fetched in memory by the $\overline{RAS}$ action and transferred into the SAM by the $\overline{DT}$ positive edge. The data of the selected bit becomes available at the next SC edge. In a write transfer, the SAM data must not change during the $\overline{RAS}$ negative edge; the clock period is hence lower than the $\overline{RAS}$ active-low time during the transfer. As a general rule, the beginning of a $\overline{RAS}$ cycle with $\overline{DT}$ active must not occur when data is shifted into the SAM.

SAM pseudowrite transfer. An interesting case occurs when a write transfer must follow a read transfer. The SAM start address is given in the transfer cycle and must precede a serial-in transfer (memory-write transfer). But the effective serial-in transfer is performed after the data is shifted. Hence, one must execute a pseudowrite transfer, which is similar to a write transfer but does not transfer the SAM data toward memory (Figure 17). This is encoded on the $\overline{SE}$ line; if this line is active low at the $\overline{RAS}$ active edge, a normal write cycle occurs. If $\overline{SE}$ is not active, a pseudowrite cycle occurs.

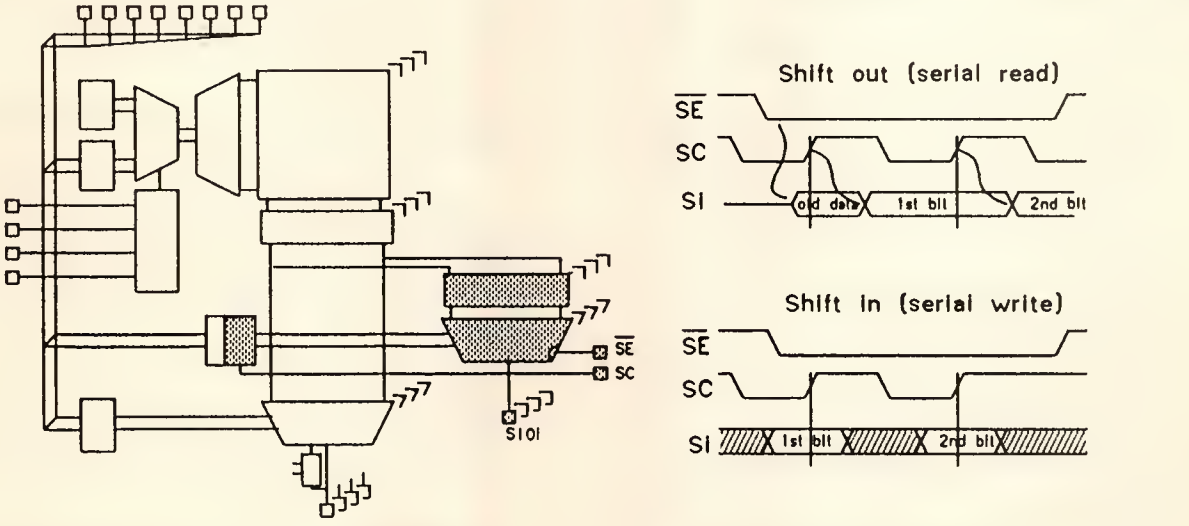


Figure 16. SAM shift timing.

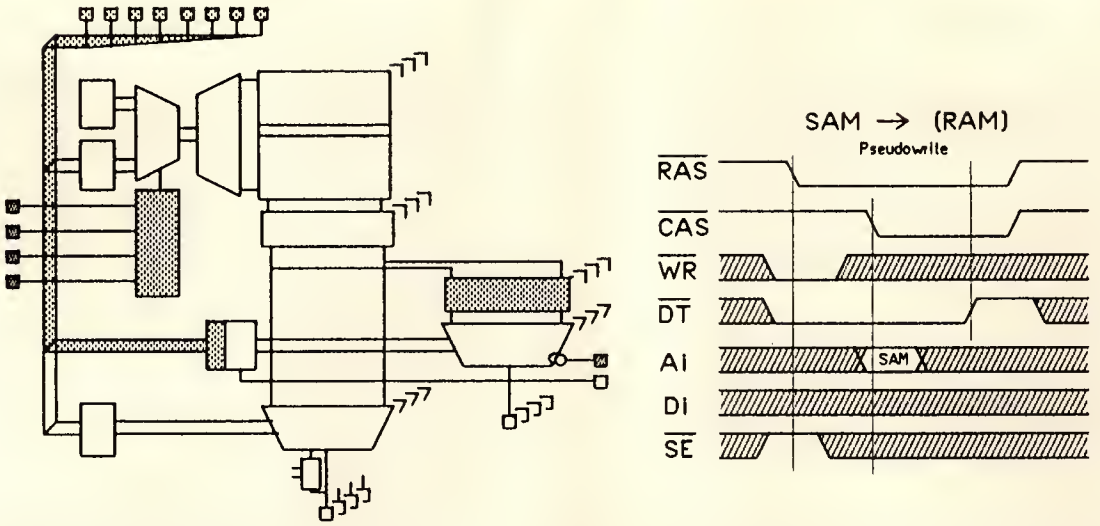


Figure 17. SAM pseudowrite transfer.

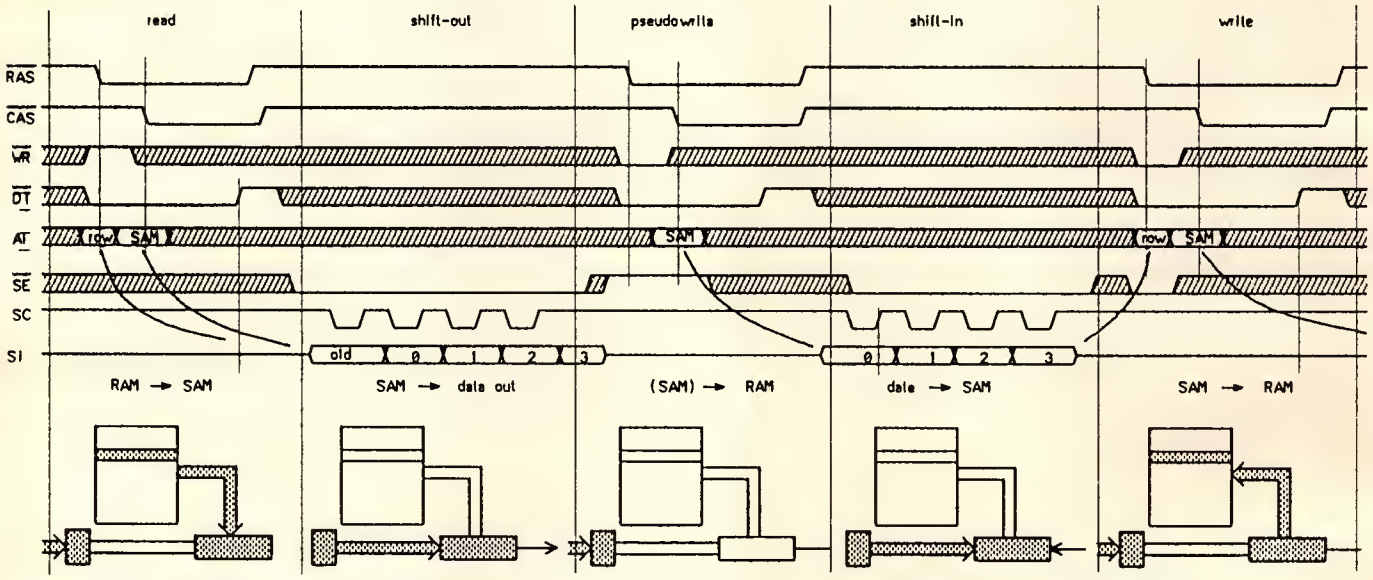


Figure 18. Example of utilization of a pseudotransfer.

Figure 18 shows a typical alternating-shift sequence. A read cycle is followed by a pseudowrite cycle to obtain a SAM address, and then by a write cycle.

Display interface using VRAMs

VRAMs have been designed for video displays and are well suited for that application. The actual memory size of 64K × 4 is adequate for screens available at present. The progressive development of TV screens is far slower than that of memories, and we expect that the 64K × 4 parts will represent the largest portion of the market for many years to come. Only high-resolution color screens need 256K × 4 parts (Figure 19). Future high-capacity VRAM devices will require new internal organization, with a wider data bus and more dedication towards given applications. The VRAM hybrid modules described later show a probable direction.

Minimum black-and-white screen. Four VRAMs are sufficient to create a high-resolution graphics screen of 1024 × 1024 dots, with a maximum bandwidth of 100 MHz. Figure 19 shows the rather simple schematic with

- only a few counters,
- a 4-bit shift register for multiplexing the 4 bits available on the serial ports of a memory chip, and
- a 4-bit decoder for selecting the serial outputs of one memory chip after the other.

This multiplexing generates a 4 × 25-MHz bit stream. A 16-bit shift register could be used, but

would require more complex wiring and a larger power dissipation.

A 24-pin PAL generates the control signals; the processor triggers the read/write cycles, and the Display Enable signal from the display controller triggers the transfer cycles during horizontal retrace. A careful design must arbitrate between the requests and avoid metastable states.

The 6845, 6345, and derivatives are among the few display controllers that can be used for graphics display applications incorporating RAMs, but they are far from being perfect. Address limitations exist, and one should be careful with the address provided by the 6845 when the RAM-SAM transfer occurs. Another problem is the blanking of the first line enabled by the 6845, since the address was not yet correct when that line was prepared.

Controlling the SAM-RAM transfers from the processor instead of from the video display controller can be very powerful. This implies duplicating the memory space for normal and SAM access or using an I/O mode bit. In this case, one can write a row in memory, copy it into the SAM, and reload it to any other row. The SC must of course not be active (no clock SC or $\overline{SE}$ disabled).

Using that feature, which implies only a better programmed PAL and few instructions, one can clear a 128K-byte bitmap (1024 × 1024 pixels) on a 16-bit bus by performing 256 write cycles (clear) on a given row, then one RAM-SAM transfer and 255 SAM-RAM transfers to all the other rows.

Bit organization. Memory organization is not evident, even on a black-and-white display. Most people will agree that pixels should be numbered starting from zero on the upper left corner (Figure 20.) Logic supports byte number increase in proportion to the pixel number. Disagreement arises

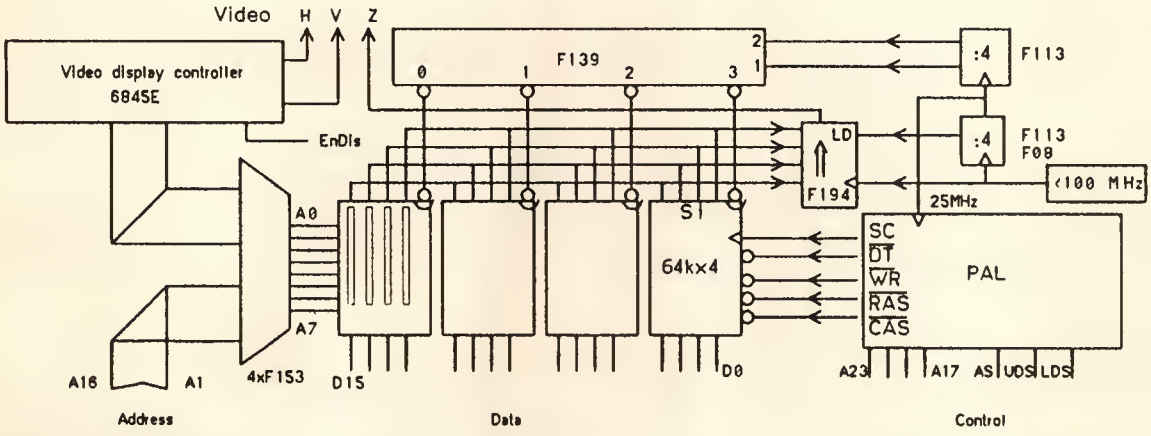


Figure 19. The 16-bit screen.

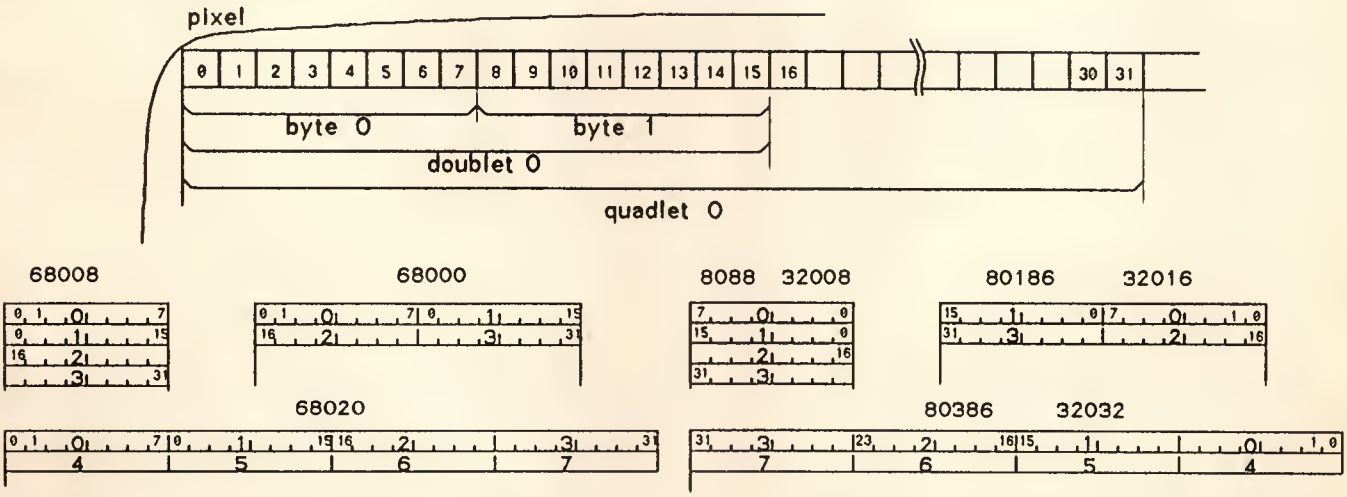


Figure 20. Bitmap organization for various processors.

with numbering bits, due to the famous "Big Endian against Little Endian fight."[10] It is evident that bit 7 (MSB) of the first byte must be pixel 0 on a memory controlled by a Motorola processor. The reverse is true for Intel and National Semiconductor, and some efficiency loss would occur if two processors such as Intel and Motorola share the same screen.

Color-screen organization. Color screens also offer several possibilities for mapping the multibit pixel into memory. Depending on the number of bits per pixel, the optimal solution may differ.

Figure 21 shows four basic solutions in the case of a 4-bit-per-pixel screen. Solution a) takes 1 byte per pixel. This is clean, but slow. Solutions b) and c) pack two nibbles in a byte or four in a doublet, and the order of both nibbles and bytes is a debatable subject. Solution d) corresponds to four superimposed monochrome screens, and the components for one pixel are found at different processor memory addresses.

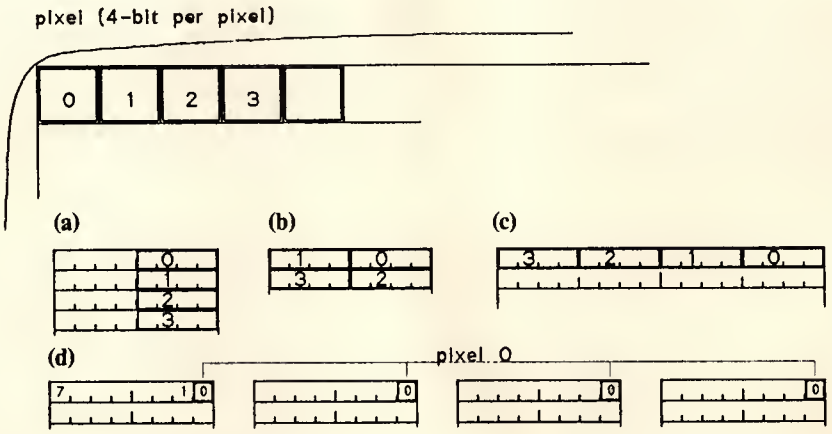


Figure 21. Color mapping in main memory, solutions (a) through (d).

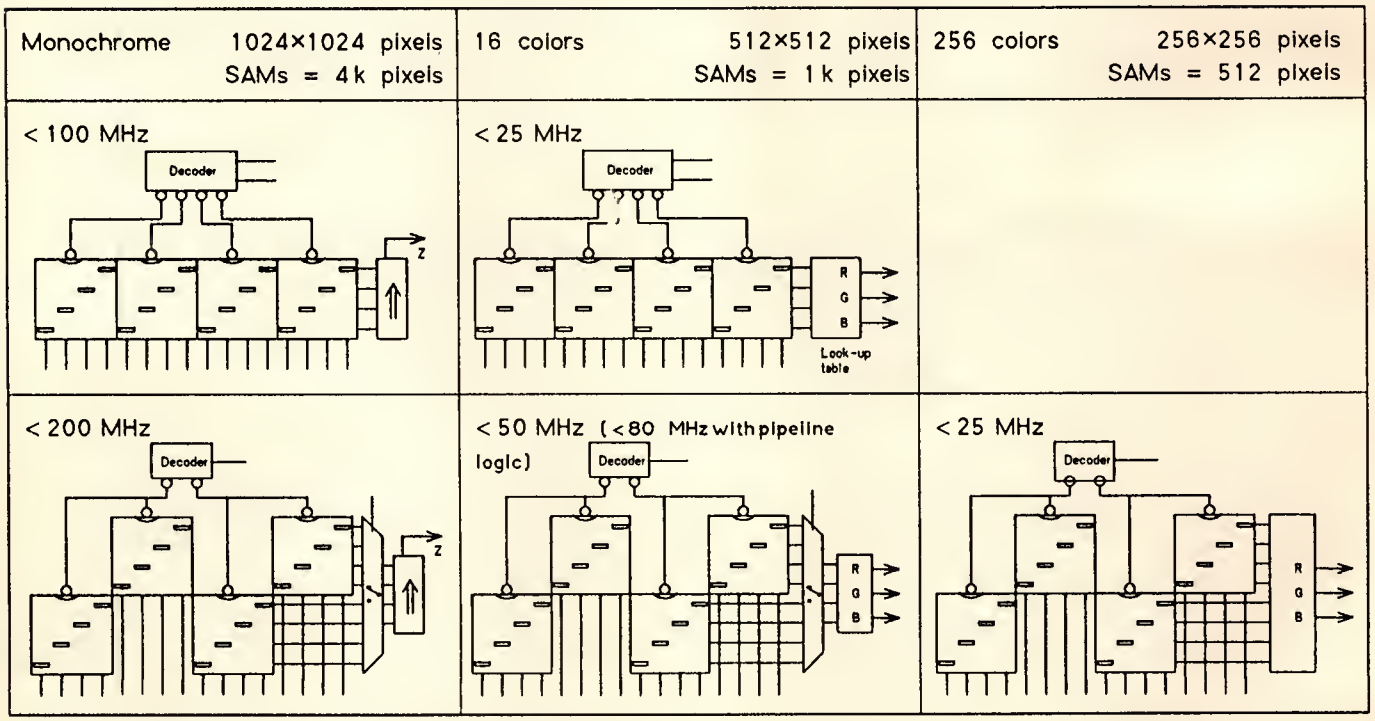


Figure 22. Displays built with four 64K × 4 VRAMs (128K bytes).

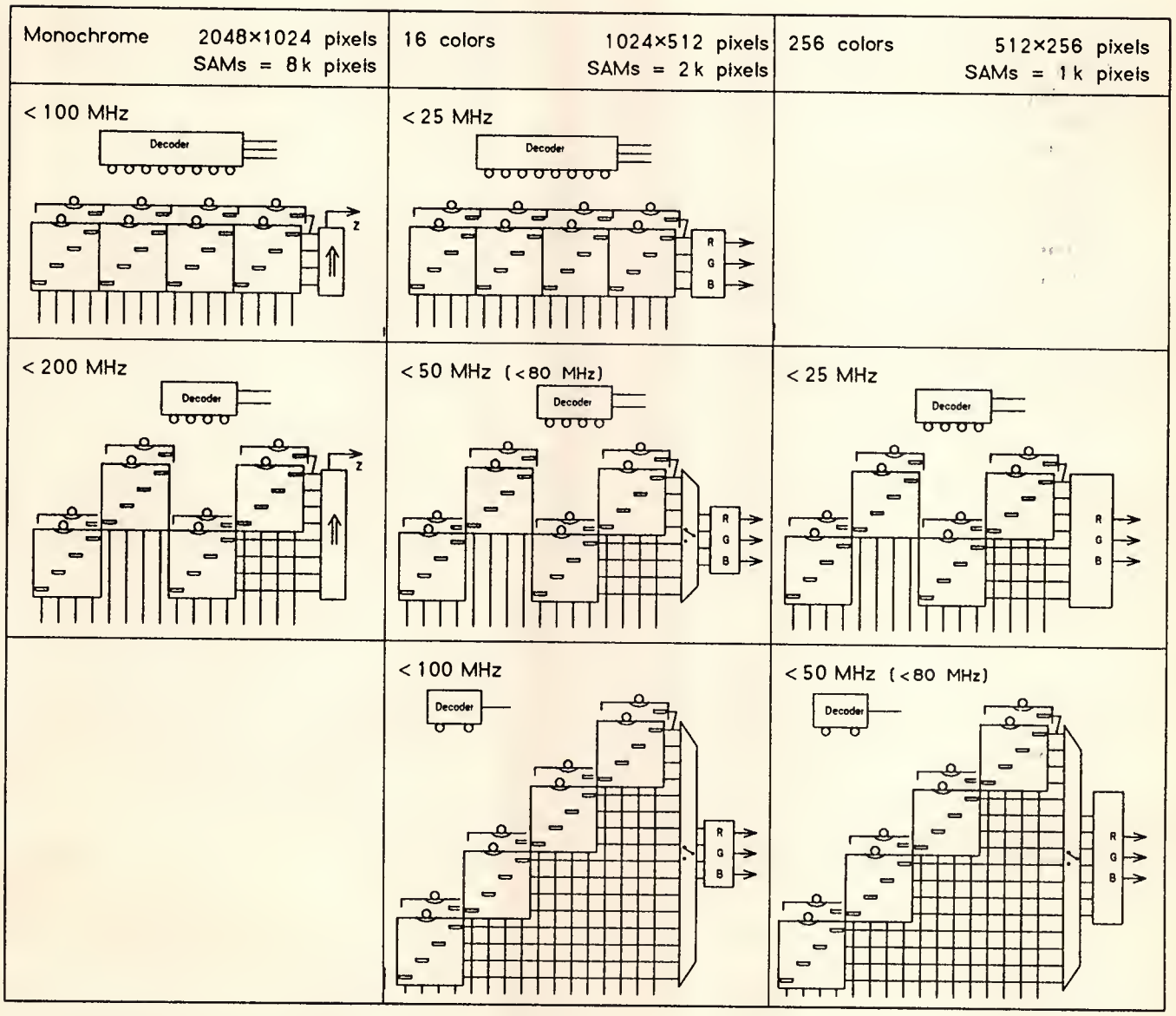


Figure 23. Displays built with eight 64K × 4 VRAMs (256K bytes).

The selection of an existing display controller usually forces the choice; very good circuits exist for CAD applications, but they are not efficient for monochrome typographic applications.

VRAM organization for different resolutions. The number of VRAMs necessary for a given display application depends on the screen resolution, the bus width, and the number of color planes. Consider the case of a 16-bit bus and the existing 64K × 4 VRAMs with a 25-MHz shift frequency. Four VRAM circuits already allow an interesting screen size (Figure 22): Each VRAM has four internal shift registers. However, the serial output of these shift registers is bused, and a decoder selects one chip after the other, as already explained in Figure 18. The capacity of the SAM register is always superior to a screen line, so there is no need to reload the SAM in the middle of the screen.

Multiplexing the output with some fast logic allows an increase in bandwidth. Additional registers and correct phasing of the clocks allow for higher speeds than are noted in the next figures.

For an increased resolution, one can use a 32-bit bus or add a second bank as in Figure 23. The addresses of the two banks can be successive or interleaved. The selection is made through the $\overline{RAS}$ and $\overline{CAS}$ lines.

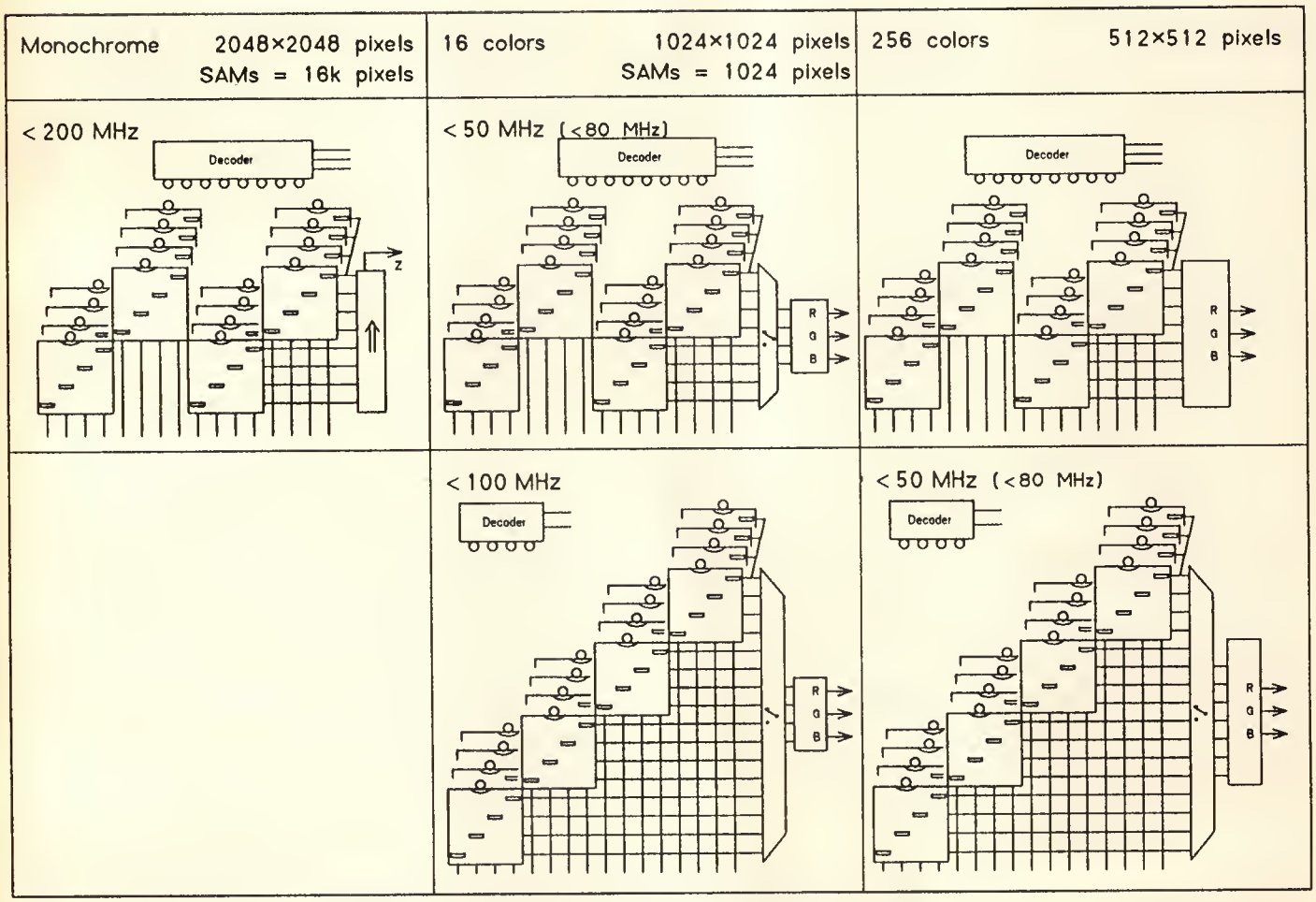


Figure 24. Displays built with sixteen 64K × 4 VRAMs (512K bytes) or four 256 × 4 VRAMs.

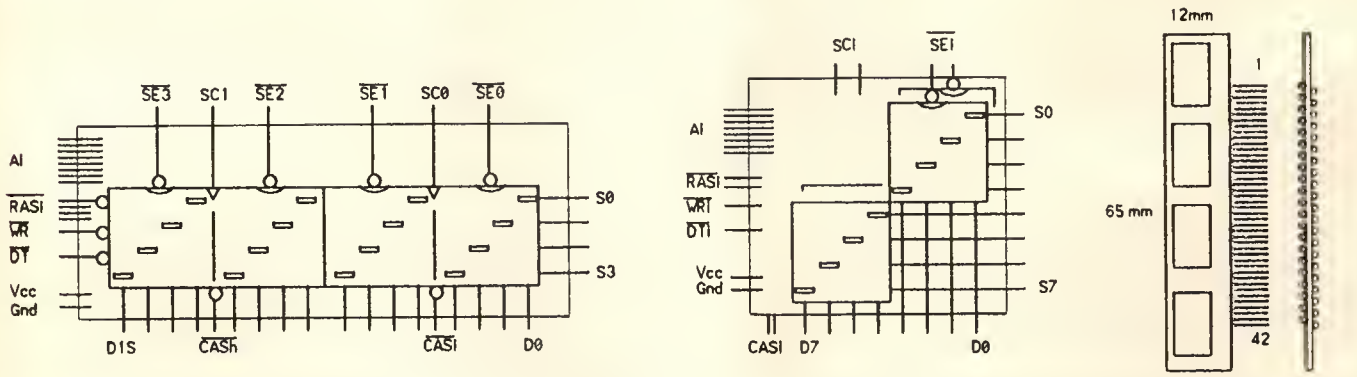


Figure 25. VRAM modules (Valtronic).

The simple linear arrangement is of no interest for very high resolution screens (Figure 24). Future 256K × 4 VRAMs will be adequate for these high resolutions. The color lookup table for 256-color resolution and the high-frequency shift register (or multiplexer) exists in a single hybrid module from Brookree Corp. (the Ramdac BT458). This module simplifies the design considerably.

VRAM modules. Hybrid modules including four VRAMs have existed for some time in order to emulate 64K × 4 by using four 64K × 1 of the

previous generation. Valtronic Inc. has developed new modules for easier implementation of displays and other systems. These modules will not be replaced by the next 256K × 4-VRAM generation. They do need a greatly reduced real estate and a pin count divided by two or more, due to the sharing of addresses and serial lines. The two modules in Figure 25 are standard and are optimized for the architectures of the previous figures. Valtronic can make others on request. Using these modules, we realized a 26-plane, 1024 × 1024 pixel-color display on a 220 × 310-mm VME board in our laboratory. (See photograph on second page of this article.)

Fast block transfers

As previously mentioned, VRAMs offer interesting possibilities for fast communication between systems. Using the usual RAM port, one can read or write a memory word in about 250 ns. By using the SAM port (and losing part of the random-access capability), this time is decreased to 40 ns (25-MHz shift frequency). At that speed, the SAM is transferred in 10 μs (256 × 40 ns), and the complete 64K × 4 memory block is transferred in 2.6 ms.

However, if the VRAM is organized as 32-bit words, it may not be cost effective to implement a 32-bit, 25-MHz connection. Multiplexing is possible with or without performance limitation. With fast TTL logic, 100 MHz can be reached; special components and fiber optics are required in the range of 160 MHz or higher.

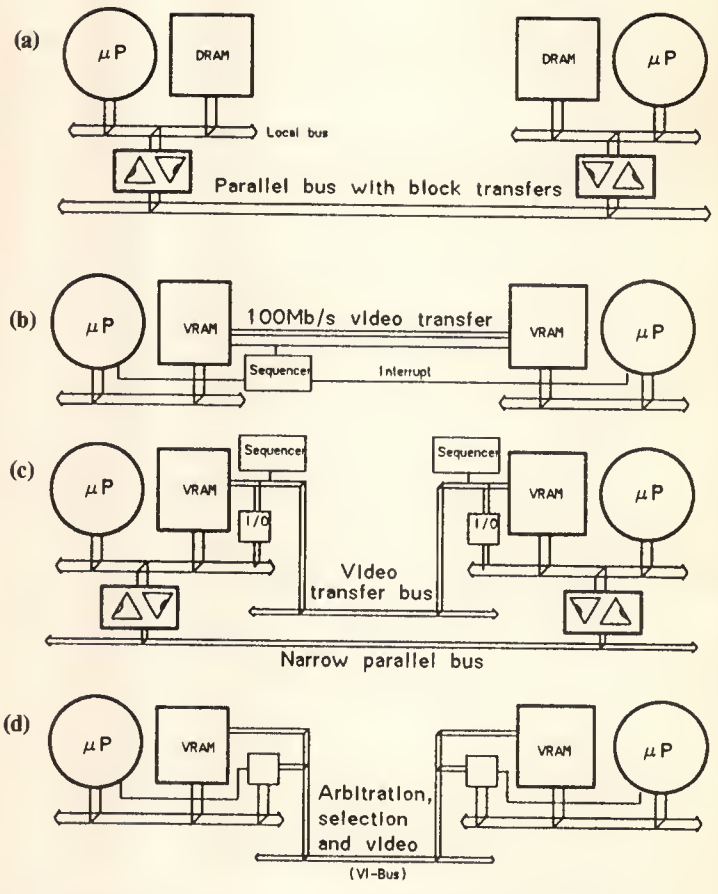


Figure 26. Basic solutions for block transfers, (a) through (d).

Multiprocessor communications. A multiprocessor system consists of several subsystems including a processor, a memory, and possible I/O. In a typical system (Figure 26a), the backplane bus (P896, VME, Multibus, Nubus, M3-bus)[11] transfers the data under the control of a processor or a DMA unit. The maximum transfer rate on a 32-bit bus is 16M bytes/s. Hence, the transfer of 1K byte at full speed takes 64 μs, but during this time both processors cannot use the bus and the main memory.

In the simple case of communication between two systems, the VRAM solution can be implemented with a logic very similar to that of the monochrome display of Figure 18. Using eight parallel lines instead of one (eight F194 TTL circuits are required) with a clock frequency of 100 MHz produces a transfer time of 10 μs for 1M byte (32 registers of 256 bits). This time is 25 times faster than the previous solutions.

This solution supposes of course that the processor does not have to move the information to be transferred into the VRAM. This procedure would cost almost as much time as the bus transfer of Figure 26a and is not worthwhile.

Simply put, the scheme of Figure 26b supposes that the transfer logic interrupts both processors when the transfer is finished. This interruption also costs about 10 μs but can be reduced if a DMA device performs the transfer of blocks of 1K each and interrupts when the correct number of blocks is reached.

A video-transfer bus must be defined to transfer the video information between more than two units. A simple solution is to use the main global bus as an addressing mechanism for defining the source and destination of the video transfer. The global bus of such an architecture does not have to be very fast, since it passes solely control information for the VRAM transfers and synchronization information (semaphores) for the coordination of the processes. A simple bus such as the Reduced Synchronous Multiprocessor bus developed at Lausanne (Resym)[12] could be used for a low-cost, efficient system.

The global bus can be avoided if the video bus includes the control lines for arbitration and selection (Figure 26d). This scheme is not worthwhile if most messages are short because of the increased amount

of logic: arbitration and selection on each access requires about 40 ns. Video transfer follows with the speed characteristics mentioned earlier, and that solution is worth it for large data blocks. The reduced number of lines of the bus allows consideration of applications with several buses to increase the parallelism within a pool of processors.

Vi-bus specifications. The video bus, or Vi-bus, which corresponds to the Figure 26d, was designed for high-speed transfer of data blocks between systems having an adequate VRAM interface. The Vi-bus provides an arbitration cycle to select the data transmitter, an addressing cycle to select the receiver(s), and one or more block data-transfer cycles. Compared with a common parallel bus, the Vi-bus executes only write transactions, with capability for broadcast cycles. The fully multiplexed Vi-bus consists of an information path of eight lines (INF0-INF7) and six control lines. The arbitration, selection, and transfer cycles execute sequentially using the same information lines, without any pipeline. This choice reduces the number of bus lines and drivers, but it increases the time required for a bus transaction. However, this increase is not a serious drawback because the arbitration and selection cycles require about 600 ns, and the data transfer for each byte takes 50 ns. If the transferred block is longer than 16 bytes, the total transaction time consists mostly of the transfer time.

The Vi-bus control signals are (an asterisk indicates an active low signal):

- Bus Clock (BCLK), a 5-MHz continuous clock used for the arbitration and selection cycles;
- Bus Busy (BB*), an open collector signal issued by the transmitter indicating that a transaction is occurring on Vi-bus;
- Data Enable (DE*), a 3-state signal issued by the transmitter indicating that a data-block transfer is terminated;
- Wait (WAIT*), an open collector signal issued by the receiver that is the block synchronization signal;
- Strobe (STB), a 3-state signal issued by the transmitter whose leading edge validates the data on INF lines; and
- Interrupt (INT*), an interrupt request signal for all the processors which is mainly used for debugging and performance measurement.

The arbitration cycle, synchronous with the BCLK, takes one clock period. The arbitration is obtained with a self-selection circuit. If a maximum of eight processors on the Vi-bus is allowed, the self-selection can be replaced by a linear selection.[11] In the BCLK period following the arbitration, the bus winner places the destination identifier (possibly using geographical addressing) on INF lines. In the following BCLK period it issues the address for the destination memory. The selection cycle, also synchronous, requires two BCLK periods. After this cycle the system enters the data-transfer phase in which one or more data blocks are transferred.

Each data block transfer is ruled by a semisynchronous protocol, using the DE* and WAIT* signals. The byte transfers within each block are performed with a synchronous protocol, using the STB signal. Each block has a length variable from 4 to 1024 bytes, by multiples of 4; the block end is signalled by DE* returning inactive. After each block transfer, the receiver copies its SAM into the DRAM, and the transmitter copies the following DRAM row into the SAM for the next block transmission.

The WAIT* signal guarantees the correctness of the block transfer, even if the transmitter copy operation is faster than the receiver copy operation.

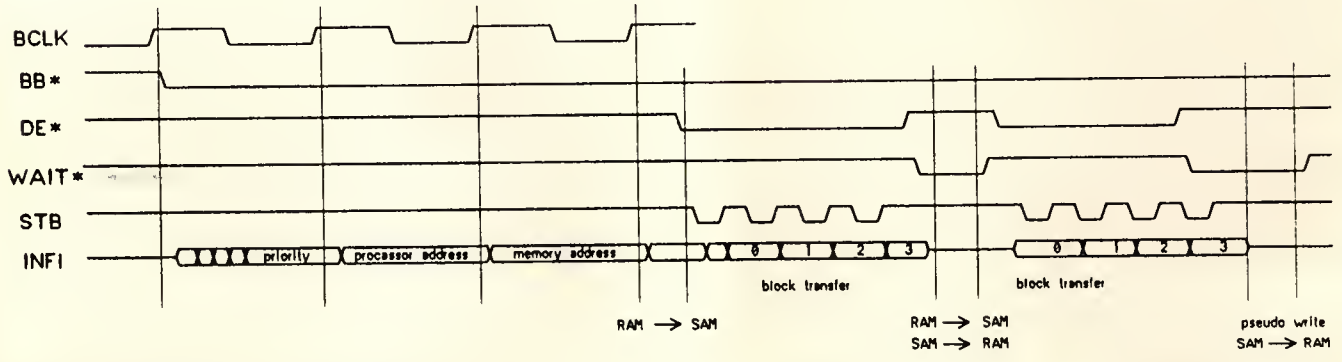


Figure 27. Typical Vi-bus transaction.

The use of an active, low, open collector line allows for broadcast operations because the slowest receiver can delay the transmission of the next block.

After the transmission of the last block, the transmitter executes a pseudowrite cycle to put the VRAM in input mode and the sequential pointer at the beginning of the SAM, and releases the bus, deactivating BB*. This signals to the receivers that the transaction is terminated. Figure 27 provides an example of Vi-bus transaction.

The video RAM is a first step toward smart memory. Its evolution may follow two directions:

- VRAMs for graphics display applications do not need to be significantly larger than what is presently available. They would benefit from a better interface with display controllers to save time and power-consuming interface circuitry using many PALs.
- VRAMs for multiprocessor applications must be as large as possible to allow the use of VRAMs as main memory. Dedicated logic could be included inside the VRAM to simplify the interface toward a bus (arbitration, control, and sequencer). This approach takes at least one more pin for interrupts; control information can be transferred using the scheme for loading the logic-function register (Figure 9). A larger 30-pin package with an 8-bit-wide bus could be very efficient for this purpose.

Acknowledgments

We thank F. Maddaleno, C. Nicks, R. Beuchat, and the referees for their corrections and suggestions. Part of the work has been supported by the Fond National Suisse (grant FN2.147-0.86) and the Commission pour l'Encouragement des Recherches Scientifiques (grant 1556n). We wish to express our thanks to both organizations.

References

1. S.H. Leibson, "Graphics-Controller ICs," *EDN*, Feb. 6, 1986, pp. 104-118.
2. E.R. Hnatek, *A User's Handbook of Semiconductor Memories*, J.Wiley & Sons, New York, 1977.
3. M. Wright, "Dynamic RAMs," *EDN*, Feb. 20, 1986, pp. 135-146.
4. D. Wooten, "Nibble-mode Technique Simplifies Complex RAM Designs," *EDN*, Feb. 17, 1983, pp. 231-242.
5. D.L. Finke, "Dynamic RAM Architectures for Graphics Applications," *Proc. NCC*, 1983, pp. 479-485.
6. J. Fallin, "CHMOS DRAMs in Graphics Applications," Application Note AP-172, Intel Corp., Santa Clara, Calif., 1984.
7. "Multi-Port Video RAM HM53461/462," Application Note 10-8, Hitachi, Muenchen, West Germany, 1987, 108 pp.
8. R. Pinkham et al., "A High Speed Dual Port Memory with Simultaneous Serial and Random Access for Video Applications," *IEEE J. Solid State Circuits*, Dec. 1984, pp. 999-1007.
9. R. Pike, "Graphics in Overlapping Bitmap Layers," *ACM Trans. Graphics*, Apr. 1983, pp. 331-341.
10. D. Cohen, "On Holy Wars and a Plea for Peace," *Computer*, Oct. 1981, pp. 48-43.
11. D. Del Corso, H. Kirrmann, and J.D. Nicoud, *Microcomputer Buses and Links*, Academic Press, London, 1986, 415 pp. (*See book review in this issue on p. 75. – Ed.*)
12. J.D. Nicoud, "Resym, a High Performance, Low Power Multiprocessor Bus, " *Proc. 13th Symp. Comp. Arch., Computer Architecture News*, 1986, pp. 169-174.



Jean-Daniel Nicoud is a professor at the Ecole Polytechnique Federale de Lausanne in Switzerland. He has been active in microprocessor-related research for 15 years and has designed many microprocessor-based systems. His interests and those of his research group include microcomputer design, development tools, local networks, microcomputer peripherals, graphics workstations, laser printers, and VLSI.

Nicoud holds a degree in engineering physics and a PhD degree in electrical engineering, both from the Ecole Polytechnique Federale de Lausanne. He was an associate editor of *IEEE Micro* from its inception in 1981 until 1985.

Questions concerning this article can be directed to the author at Ecole Polytechnique Federale de Lausanne, Laboratoire de Microinformatique (EPFL—LAMI), Av. de Cour 37, CH-1007 Lausanne, Switzerland.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 150 Medium 151 High 152

FEATURE

GaAs Microprocessors and Digital Systems

AN OVERVIEW OF R&D EFFORTS

Harry Vlahos and Veljko Milutinovic
Purdue University

Projects based on the emerging technology of GaAs are attracting interest in both scientific and academic communities.

Microprocessors manufactured with gallium arsenide produce the unusually high speeds demanded for implementing very complex applications such as signal processing and supercomputing. These chips perform well in extreme environments, making them especially useful for vital military applications. Eventually, we expect to see GaAs chips extended to a variety of commercial markets.

Here we provide an overview of existing gallium arsenide microprocessors and related digital designs. For more information on GaAs technology itself, see Larson et al.[1]

The architectures we describe cover a wide range of applications and were chosen on the basis of their unique design and technology characteristics. Some important work in GaAs processor design, however, could not be covered due to the lack of public-domain information, or simply because the projects are still in their early phases. Relevant issues for this article relate to the processor-design level with some register-transfer-level components covered for the sake of completeness.

We chose a level of presentation detail that we believe is enough for readers to understand the essence of each project. Only some noteworthy low-level details are covered. Complete lower level details on each project appear in publications listed in the references. Readers should achieve the maximum benefit with this approach, because design details that may hide the essence of the project, or represent common knowledge, are avoided. The box on the next page lists often-used abbreviations.

Bit-slice and related components

Bit-slice components permit the design of flexible systems that overcome many of the deficiencies of a conventional microprocessor: limited word lengths, inflexible instruction sets, and predefined architectures. The control and data processing functions of a bit-slice-based microprocessor are manufactured on separate chips and can be configured in various ways to obtain unique architectures with varying word lengths. Since users can microprogram these bit-slice machines, instruction sets can be optimized for particular applications. For general information on bit-slice systems refer to Mick and Brick[2] or White.[3]

Bit-slice components. Two companies that have already developed GaAs versions of a subset of the Am29xx family of bit-slice components are Vitesse Electronics Corporation and McDonnell Douglas

0272-1732/88/0200-0028$01.00 © 1988 IEEE

Corporation. Their GaAs designs are functional and microcode compatible with the silicon versions from Advanced Micro Devices (AMD).

Vitesse Electronics 2900 series. Vitesse Electronics Corporation (VE), located in Camarillo, California, set out to design GaAs versions of the industry-standard components from Advanced Micro Devices. Through an agreement with AMD, VE reimplemented in GaAs the Am2901 4-bit microprocessor, the Am2902 carry-lookahead generator, and the Am2910 microcontroller. Every member of the VE29G00 family is I/O compatible with standard (100K) emitter-coupled logic (ECL), allowing designers to work with a familiar set of specifications.

The speed of the GaAs processor may be as much as four times that of its silicon equivalent. For example, the minimum cycle time of the control loop for a 16-bit processor constructed from four 4-bit slices using VE29G00 parts is 22 nanoseconds. The equivalent system using bipolar ECL Am2900C parts takes 98 ns. Data loop times are 29 ns and 97 ns, respectively, for the VE29G00 and the Am2900C parts.

The components produced by Vitesse Electronics are function- and microcode-compatible with the silicon versions from AMD. However, these parts are not drop-in replacements for Am2900 components. The GaAs systems must be carefully designed to optimize off-chip path lengths, clock skews, and other speed-related parameters that may degrade the extra processing power available through GaAs chips.[4]

The VE29G01 is a high-speed, cascadable, 4-bit arithmetic logic unit, or ALU, fabricated using enhancement/depletion-mode metal semiconductor field-effect transistor, or E/D MESFET, GaAs technology.[5] The device consists of a 16-word-by-4-bit, dual-port RAM; a high-speed 4-bit ALU; shifting logic; decoding; and multiplexing circuitry. For readers unfamiliar with the internal structure of the Am2901 ALU, we present the basic facts here. The block diagram of the VE29G01 is identical to that of the Am2901.

The chip has a cycle time of 14 ns from the selection of the A and B registers to the end of the read-modify-write cycle. This time compares favorably to the 32-ns cycle time required by the silicon version. It contains approximately 500 logic gates composed of over 5000 transistors.

The VE29G10A microprogram controller is a high-speed microinstruction sequencer fabricated using GaAs E/D MESFET technology.[6] It controls the sequence of microinstructions stored in a microprogram memory with a maximum size of 4K microwords. VE29G10A capabilities include conditional branching, subroutine return linkage, and looping with a maximum count of 4096 iterations. The microprogram controller, actually an "intelligent multiplexer," provides a 12-bit address from one of four sources:

- the microprogram address register,
- an external direct input,
- a register/counter, or
- a nine-word-deep LIFO stack.

The GaAs chip has a minimum clock low time of 6 ns and a minimum clock high time of 6 ns. This translates to a minimum clock period of 12 ns, approximately 10 times faster than the equivalent silicon version.

McDonnell Douglas 2900 series. The Am2901-equivalent component from McDonnell Douglas (the MD2901 bit-slice ALU) is functionally equivalent to the VE29G01 just described.

The MD2901 also emulates the operation of the popular Am2901A, which has been widely used in signal processing applications. The initial chip, successfully fabricated using a GaAs enhancement-mode junction field-effect-transistor, or E-JFET, process, has demonstrated an operating speed of 25 MHz with a power consumption of only 135 mW.[7] McDonnell Douglas is working on an improved version of the MD2901 to increase the speed dramatically. One of the changes includes a new layout that minimizes the size of signal and bus lines. Future implementations of the MD2901 are expected to be operating at speeds up to 100 MHz.

Definition of Terms

BFL: Buffered FET logic
CISC: Complex instruction set computer
DCFL: Direct-coupled FET logic
ECL: Emitter-coupled logic
E/D MESFET: Enhancement/depletion-mode metal semiconductor FET
E-JFET: Enhancement-mode junction FET
FET: Field-effect transistor
ISA: Instruction set architecture
LSI: Large-scale integration
MODFET: Modulation doped FET
MSI: Medium-scale integration
RISC: Reduced instruction set computer
SACSET: Sidewall-assisted, closely spaced electrode technology
SDFL: Schottky diode FET logic
SSI: Small-scale integration
TTL: Transistor-transistor logic

An Am2910 equivalent in GaAs is still under development at McDonnell Douglas. Therefore, no additional information is available on this component.

Memory chips. In recent years, numerous companies both in the US and Japan have devoted a great deal of effort in the design and fabrication of GaAs memory chips to be incorporated in GaAs microprocessor systems. These chips are typically used for cache memories. Successful fabrication of chips with sizes of 1K to 4K bits have been achieved by many companies.

We selected two very special research efforts. One from Nippon Telephone and Telegraph has surpassed, in circuit density and memory size, many competitors' designs. The other effort from Purdue University resulted in the development of the first one-transistor, dynamic RAM cell in GaAs.

Static random access memory. Efforts from companies in both the United States and Japan have proven successful in the development of static RAMs, or SRAMs, with capacities of 1K to 4K bits. Due to their low access times, these chips have been incorporated into high-speed cache memory designs for GaAs processor systems.

Nippon Telegraph and Telephone Corporation (NTT) in Japan has successfully fabricated, using E/D MESFET technology, a 16K-bit SRAM consisting of more than 100,000 transistors on a single chip.[8] Current GaAs technology allows (for reliable mass production) only about 30,000 transistors on a chip without decreasing the yield considerably. The NTT achievement has surpassed one of the barriers of GaAs chip design and is not expected to be matched by competitors for several years.

The 16K-bit SRAM wafer, however, suffers from a high defect density, which decreases the yield to about 1 percent, even if state-of-the-art fabrication technology is used. If the defect density could be reduced to 10^{-2}cm, a 16K-bit SRAM could be produced with a yield of about 10 percent.

NTT fabricated the chip using E/D MESFET direct-coupled FET logic (DCFL) GaAs technology with the conventional six-transistor memory cell. The organization of the chip is 4096 words by 4 bits. The access time, measured to be 4.1 ns, is comparable to other GaAs SRAMs with lower circuit densities.

Dynamic random access memory. To date, the only memory elements available to GaAs system designers were six-transistor static RAMs. These require a considerable chip area and consume power continuously.

Professors Cooper and Melloch and their associates at Purdue University have developed a one-transistor dynamic RAM cell for use in GaAs ICs. Initially designed with aluminum gallium arsenide/GaAs (AlGaAs/GaAs) heterojunction FET technology, the chip demonstrated storage times of 75 seconds at 175°kelvin, 500 minutes at 140°K, and 220 hours at 77°K.[9]

Cooper and Melloch's recent work in this area has yielded even more interesting results. A GaAs, buried-well, dynamic RAM cell operating at a room temperature of 25° Celsius has demonstrated a 170-second charge recovery time constant, which decreases to 85 milliseconds at 125 degrees.[10] An AlGaAs heterostructure produced a 1.36-hour charge recovery time at room temperature, which decreased to 680 ms at 125°C. Extrapolation of these storage times to 77°K (liquid nitrogen temperatures) yields an expected storage time of 10^{21} seconds, which (for most practical purposes) may be considered static. The observed capacitance-recovery time constants indicate that such a structure could be used to develop modulation doped FET- (MODFET) or MESFET-compatible single-cell memories, which could operate dynamically at room temperatures, or statically at 77°K.

SSI/MSI chips. Several companies have already introduced GaAs digital ICs with gate complexities on the levels of small-scale and medium-scale integration (SSI/MSI) that may be used for the design of a complete computer system in GaAs. Many of these digital ICs are compatible with standard ECL logic output levels that provide an easy interface to ECL components.

Gigabit digital ICs. Gigabit Logic Corporation is a company dedicated exclusively to the design and fabrication of GaAs digital ICs. Gigabit currently has the following components available in GaAs:

- a quad 3-input NOR gate (155-picosecond propagation delay),
- a dual 1-to-4 fan-out buffer (50-ps skew, 3-GHz operation),
- dual D flip-flops, edge-triggered (30-ps jitter, 3-GHz operation),
- a two-stage ripple counter with divide-by-2 and divide-by-4 outputs (3-GHz operation),
- a seven-stage ripple counter (4-GHz operation), and
- a variable modulus divider (2-GHz operation).

Current Gigabit development projects include additional digital ICs with complexities on the SSI/MSI levels. Gigabit's foundry is open to the public domain and commercial markets.

TriQuint digital ICs. TriQuint Semiconductor, a subsidiary of Tektronix, also has several high-speed digital ICs that are ECL input- and output-

compatible. Some of the components available through TriQuint are:

- a 4-bit ripple counter (3-GHz operation),
- a 4-bit synchronous up/down counter (1-GHz operation),
- a divide-by-4/5, dual modulus divider (2-GHz operation),
- a programmable dual modulus divider (divide by: 40/41, 64/65, 80/81; 2-GHz operation),
- an 8:1/16:1 multiplexer (2-GHz operation),
- a 4 1:8/1:16 demultiplexer (2-GHz operation),
- a 4 4:1 multiplexer (2.5-GHz operation),
- a 1:4 demultiplexer (2.5-GHz operation), and
- dual D flip-flops (2-GHz operation).

Other SSI-level components are currently under development. TriQuint's foundry is also open to the public domain and commercial markets.

Parallel multiplier chips. Multiplier speed is becoming increasingly important in the field of digital processing systems. Fast multipliers are essential to systems whose application is very multiply intensive, because the system-processing speed becomes limited by the speed of the multiply function.

Recently, many GaAs parallel multiplier chips have been introduced. We discuss two different approaches to the design of a multiplier in GaAs: an architecture that uses an array of adders and an architecture that implements Booth's algorithm to perform the multiplication.

Fujitsu parallel multiplier. Parallel array architectures implement fast multiply circuits using full-adder cells as the basic building block of the multiplier chip. Designers adopt these architectures because of their highly repetitive layouts. The 16-bit-by-16-bit GaAs parallel multiplier designed by Fujitsu uses this parallel array architecture instead of adopting a special multiply algorithm.[11]

Fabricated using E/D MESFET GaAs technology, the device consists of 224 full-adders, 16 half-adders, 256 NOR gates, and 64 I/O buffers, for a total of 3168 gates.

This multiplier demonstrates multiply times of 10.5 ns, at a power dissipation of 952 mW. If the FET gate length can be reduced, a multiply time of 6.5 ns at 1.6-W dissipation is easily achievable.

NEC parallel multiplier. NEC Corporation designed a high-speed, 12-bit-by-12-bit expandable parallel multiplier. It was fabricated using a novel GaAs large-scale integration (LSI) processing technology called sidewall-assisted, closely spaced electrode technology, or SACSET. This multiplier is unique in that it deviates from the traditional GaAs multiplier designs which incorporate an array of adders to perform the multiply operation. The designers at NEC chose to design the multiplier using nonarray logic and have successfully fabricated it using only 1083 gates.[12]

Booth's algorithm was used to generate the partial products, and Wallace's tree scheme was used to add these partial products and generate the result. Inputs to the chip consist of a 24-bit multiplicand and a 13-bit multiplier. The extra bits in the multiplicand and the multiplier are furnished to facilitate expansion.

With a gate delay of 170 ps, the parallel multiplier was able to perform a multiply in 4 ns, with a total power dissipation of 2.5 W. This performance is believed to be the top for nonarray GaAs logic LSIs.

Gate arrays

Gate array technology has been successfully used for semicustom LSI development. Gate arrays can replace anywhere from five to 50 SSI/MSI components and thus decrease the physical size, power consumption, and total cost of the complete system. The decreased number of off-chip interconnects makes gate arrays an excellent component for use in GaAs systems. We present two gate array designs here to show the recent advances in this technology.

Honeywell gate array. To satisfy the high demand for fast gate arrays in semicustom IC development, engineers at Honeywell designed and fabricated a GaAs gate array with on-chip RAM using Schottky diode FET logic, or SDFL, technology.[13] This particular gate array features 432 programmable SDFL cells, 32 programmable interface I/O buffers, and four 4-bit-by-4-bit SRAM on-chip memories. Each SDFL cell can be programmed as a NOR gate with as many as eight inputs or configured as a dual OR-NAND gate with four inputs each. The I/O buffers can be programmed to interface with ECL, CMOS, TTL, or SDFL logic families.[14]

This gate array is so flexible, in terms of logic programmability and routing, that the expected cell utilization is around 80 percent. It has demonstrated gate delays of 150 ps with a 1.5-mW power dissipation per gate. The equivalent gate count for this 432-cell gate array is 1296, when defined as OR-NAND logic with an equivalent cell gate count of three.

TriQuint gate array. TriQuint Semiconductor designed a configurable-cell GaAs array fabricated using E/D MESFET technology. It is a 3000-gate-equivalent array that has 64 I/O signals and handles data or clock rates of up to 700 MHz. The gate array can interface with ECL, CMOS, and TTL logic families.

The core of the array contains 1020 compound logic cells that include a cell circuit configured with E/D buffered FET logic (BFL) and designed to

optimize speed, noise margin, and parametric yield. Two NOR functions with a maximum fan-in of three can be "wire-OR'd" within a single cell to create more complex gates. The NOR-gate macro demonstrates delay times of 120 ps for unloaded configurations and 285 ps for a wire load of 3 millimeters and a fan-out of three. The nominal incremental delay per fan-out is 19.5 ps. The intercell delay is 41 ps/mm, the lowest value seen to date for any silicon or GaAs array.

General-application microprocessors

In 1984 The US Department of Defense Advanced Research Projects Agency, DARPA, awarded three contracts for the design and development of a 32-bit GaAs microprocessor and a floating-point coprocessor. The recipients of these contracts were Control Data Corporation/Texas Instruments, McDonnell Douglas Corporation, and RCA Corporation. Two of these contracts continue under DARPA support, and the third one continues under company funding.

The Stanford University MIPS (Microprocessor without Interlocked Pipeline Stages) architecture was selected to serve as a baseline for the development of these 32-bit GaAs microprocessors. Project requirements specified a clock frequency of 200 MHz and a transistor count limit of 30,000.[15] The coprocessor also had to be designed and developed with the same transistor count limit. The floating-point coprocessor was intended to accompany the microprocessor in high-speed floating-point operations.

Led by Thomas Gross, research teams from Carnegie Mellon University, Stanford University, Mayo Foundation, Control Data Corporation, Sperry Univac, General Electric, RCA Corporation, McDonnell Douglas Corporation, Texas Instruments, and DARPA developed a US Department of Defense Standard MIPS instruction set architecture. This ISA consists of 69 assembly-level instructions, an improved and modified version of the 1981 Stanford University MIPS instruction set.[16] The standard is referred to as "core MIPS." The GaAs microprocessor development teams mentioned above have designed their microprocessors to execute this standardized ISA. In addition, three high-level-language compilers (for Pascal, Lisp, and Ada) are currently being developed for this new MIPS ISA standard.

Despite the trend toward 16-bit and 32-bit computing systems, many applications find sufficient computing power and precision through simple 8-bit microprocessors. Their simple design also leads companies to first develop an 8-bit test case before proceeding with the design and development of 16-bit or 32-bit microprocessors.

Rockwell International is currently working on the development of an 8-bit GaAs microprocessor. At this point in time, very little public-domain information is available. However, RCA Corporation has developed an 8-bit microprocessor as a test case for GaAs technology demonstration.

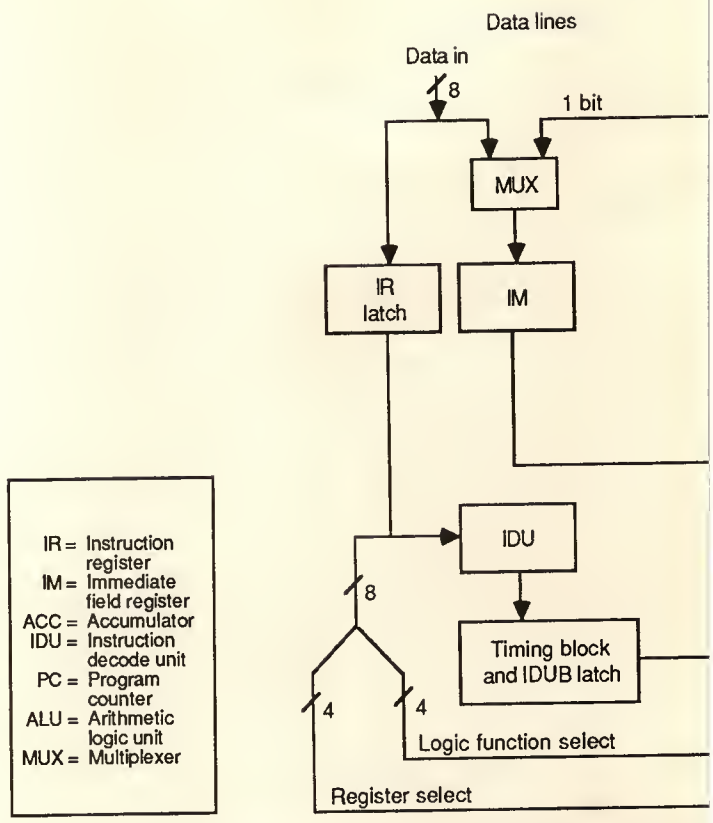
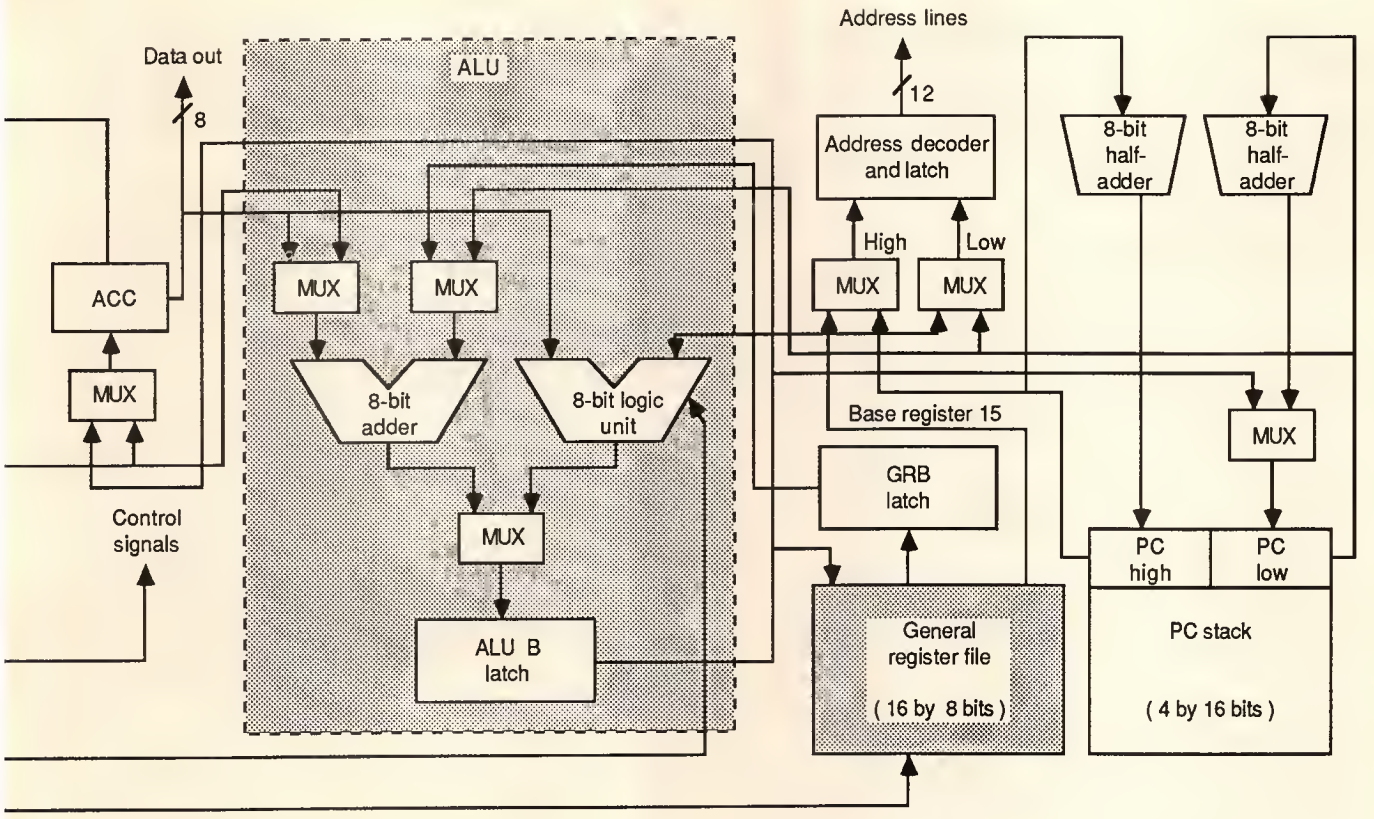


Figure 1. CPU block diagram of the Gallium Arsenide Technology

RCA 8-bit microprocessor. The 8-bit GaAs Technology Demonstration Microprocessor, or GTDM, system is a pipelined reduced instruction set computer, or RISC, machine developed by RCA as a test case for technology development and demonstration on the 32-bit CPU program.[17] Originally designed for 200-MHz operation with a performance level of 100 million instructions per second, it was later redesigned with a 200-MIPS performance goal. (We describe the 100-MIPS version here.)

The GTDM system contains 512 words of memory and minimum I/O capability that makes the system complete. Glue chips had to be incorporated for the final integration of the system components. These glue chips were constructed using ECL and leadless ceramic chip carriers mounted on a single alumina board.

CPU architecture. The GTDM CPU has an 8-bit data bus and a 16-bit address bus that allows addressing of up to 64K bytes of memory. The general register file is composed of sixteen 8-bit registers. The ALU has an 8-bit ripple-carry adder, an 8-bit arithmetic and logic unit, and an 8-bit output latch. All two-operand arithmetic and logic functions take



Demonstration Microprocessor.

one operand from the general register file and the other from the accumulator.

The branching is all program counter-relative with the 8-bit offset extracted from the 16-bit instructions. The PC stack exists only to save the return addresses from subroutine calls. Since there are only four PC registers, the nesting depth of subroutines is limited to three.

The load and store instructions use addresses that reside in the general register file. The eight most significant bits of the address are taken from register 15, and the eight least significant bits are taken from any one of the other general registers.

The CPU pipeline contains six pipestages of unequal duration. These stages are: request instruction (2.5 ns), wait (5.0 ns), receive instruction (2.5 ns), decode instruction (5.0 ns), execute instruction (2.5 ns), and write back result (5.0 ns). A new instruction is requested every 10 ns. With the overlapped fetching, decoding, and execution, as many as three instructions can be processed at one time.

Instruction decoding. The instruction set contains 23 instructions, 19 of which are 8 bits long, and four of which are 16 bits long. After an instruction is loaded into the instruction register (IR), the lower four bits are sent to the register file as a register address, and the upper four bits are sent to the logic unit to choose one of 16 possible functions. Since there are only six logic functions, the output of the logic unit is discarded if the four bits do not correspond to the code for one of the six logic functions. Likewise, if the instruction does not require an operand from the register file, the output of the register file is also discarded. All instructions fall into one of three general categories:

- two-operand, logic functions,
- two-operand, nonlogic functions, and
- non-two-operand, nonlogic functions.

Only six logic functions require two operands. These instructions use the ALU and require the output of the register file. For two-operand, nonlogic functions, the output of the logic unit is discarded. The non-two-operand, nonlogic functions (four 16-bit instructions) use an 8-bit field as a constant or branch offset and discard the output of both the register file and the logic unit. The four 16-bit instructions are decoded in the same manner as the 8-bit instructions, but the second 8 bits are destined for the immediate field, or IM, register instead of the IR register. Figure 1 is a block diagram of RCA's 8-bit GaAs CPU.

CDC 32-bit microprocessor. The GaAs Microprocessor System built by Control Data Corporation and Texas Instruments executes high-level-language programs with high performance. The system is optimized for general-purpose applications running in a virtual memory hierarchy and a multiprogramming environment.

The complete system consists of a CPU chip, a floating-point coprocessor, two memory management units (MMU), two processor board interface chips, and several RAM and support chips utilized as cache memories. The system has a full 32-bit data path and supports a 64M-byte virtual and real address space.[18]

The CPU is the heart of the system and generates all memory addresses for both instructions and data. The MMU supports a fully segmented and paged virtual memory system. The I/O processor and the CPU have separate ports on the central memory control board to a multibank central memory.

CPU architecture. The CPU designed by Control Data Corporation has dedicated buses for operand and instruction memory access. The memory interface consists of a 24-bit instruction address bus, a 26-bit operand address bus, and two 32-bit data buses: one for instructions and the other for operands. Operands are addressable in either bytes, half-words, or words, whereas instructions are only word addressable.

The general register file contains sixteen 32-bit registers, two read ports, and one write port.

The data path consists of the register file, the temporary registers, the ALU, the processor status word (PSW), and the program counters (see Figure 2). Data flows through the data path from the register file through the ALU and back to the register file.

The instruction address passes from one program counter to the next in each pipestage. The PCP1, PCM1, and PCLST program counters (see Figure 2) hold the addresses of the last three instructions so that execution after a return from an interrupt or trap can be successfully resumed. The PCNXT holds the address of the next instruction to be executed.

Pipeline design. The GaAs Microprocessor System was initially designed with a four-stage pipeline consisting of the following pipestages: instruction fetch (IF), ALU execute (EX), memory access (MEM), and write register file (WR). Since the cache memory could not support a 5-ns memory cycle required by the four-stage pipeline, the pipeline was redesigned in six stages. These stages are: instruction fetch cycle 1 (I1), instruction fetch cycle 2 (I2), ALU execute (EX), memory access cycle 1 (M1), memory access cycle 2 (M2), and write register file (WR). The memory chips have integrated on-chip pipeline registers along with the memory array to support the pipelined memory access.

During the I1 and I2 stages, the system fetches an instruction from memory by using the address in the PCNXT. The instruction address is sent to memory during the I1 stage, and the instruction is returned to the CPU during the I2 stage.

During the EX stage, the register file is read onto buses A and B. The data then passes through the ALU; the result is latched by the Result1 register (for ALU or memory instructions), sent to the PCNXT (for control transfer instructions), or latched by the processor status word (PSW) (for an implicit-move-to-the-PSW instruction).

During the M1 stage the Result1 contents are sent to Result2. The Result1 contents can be used as an address for an operand cache access. Condition codes are also evaluated during M1 and are latched into the PSW.

During the M2 stage the Result3 register latches the contents of the Result2 register, or the ALU output. If an operand access was initiated during M1, the data latches into the Result3 during this stage.

Finally, in the WR stage the contents of the Result3 register are written to the register file.

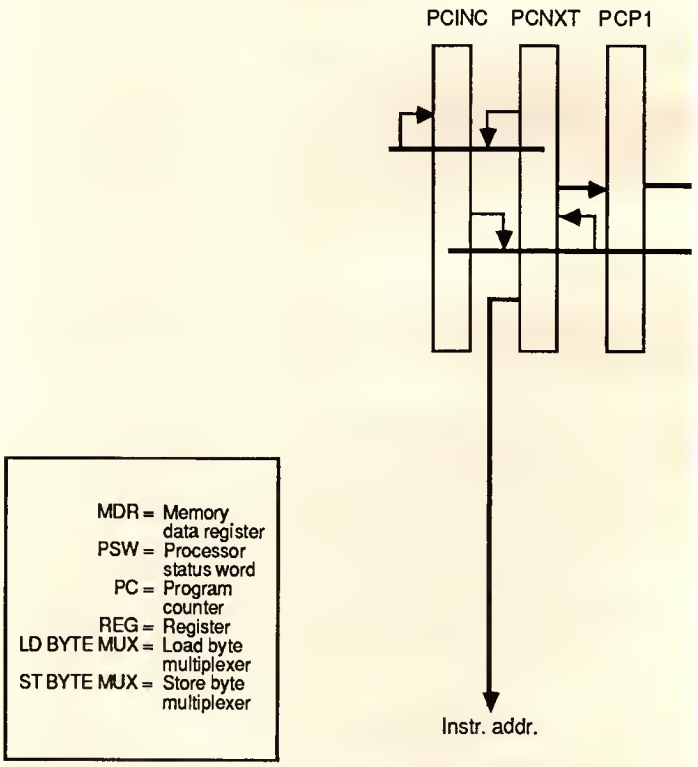


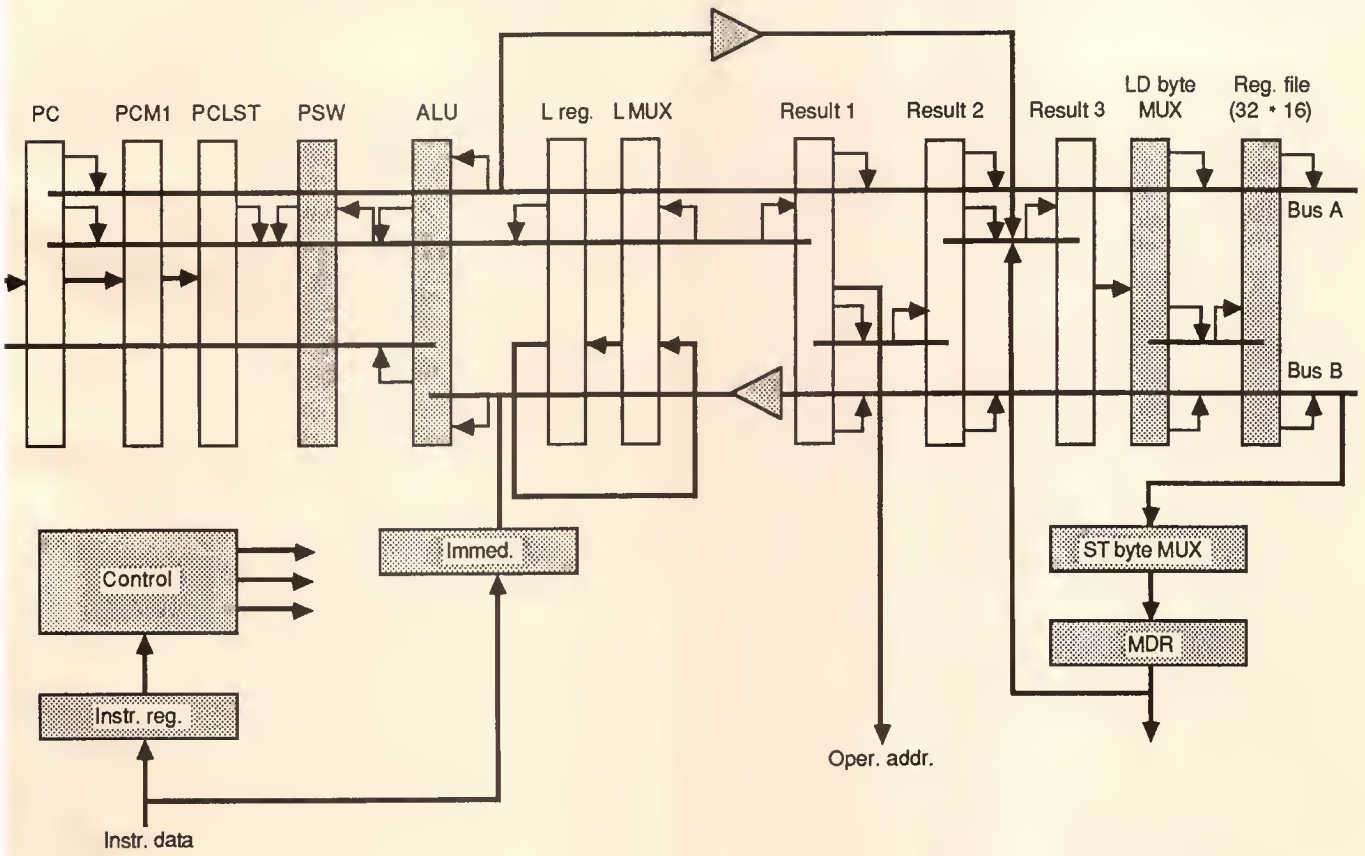
Figure 2. CDC 32-bit CPU data path.

The GaAs Microprocessor System has some hardware pipeline interlocks, in spite of the fact that the SU-MIPS baseline did not include any. However, for faster execution, only software interlocks should be used to fill in the instruction slots after a branch.

The number of branch delays for this six-stage pipeline is two, which means that two instructions that do not affect the outcome of the branch must be found to fill the branch delay slots. If no such instructions can be found, one or two NO-OP instructions may be used immediately following the branch instruction. Simulations done at Control Data Corporation show that the net MIPS rate for the six-stage pipeline is 39-percent better than for the four-stage pipeline (for selected DARPA-specified benchmarks), despite the difficulty of filling two branch delays with useful instructions (rather than just one).[18]

Cache memory. The CPU requires a 32-bit instruction to be fetched every 5 ns from the instruction memory. Furthermore, one third of all executed instructions, for DARPA-specified benchmarks, are loads or stores accessing operands in memory. To service the high frequency of instruction and operand fetches, a two-port memory is required. However, contention for the same memory locations would reduce the cache effectiveness. For this reason, designers incorporated separate instruction and operand caches to allow parallel accessing of instructions and operands without contention. A separate MMU, designed to be accessed in two pipeline stages, controls each cache. Separate address and data paths to each cache facilitate the parallel accessing.

On a cache miss, the MMU and processor interface (PI) fetch an 8-word block from main memory. During the cache fetch, the CPU remains in a wait state. The cache contains 128 single-element sets of eight words each. In a direct-mapped technique, bits 5 to 11 of the virtual address are used as indices to the tag memory sets, which contain the top 14 bits of a virtual address. The tag entry is then compared with the high-order 14 bits of the virtual address to determine a hit or miss condition. Bits 2 to 4, along with the tag-entry address bits, read the proper word



from data memory. If a hit condition occurs, the data word passes on the data bus from the cache to its destination.

Floating-point coprocessor. The floating-point coprocessor contains hardware to execute the floating-point portion of the instruction set. It follows the IEEE standard for single-precision and double-precision floating-point arithmetic.[19]

The coprocessor receives instructions and operands over two dedicated 32-bit buses. Its control interface contains the status inputs, interrupt signals, memory wait conditions, coprocessor conditions, and two signals to indicate the binary code of a particular floating-point coprocessor, if more than one is used in the system.

The data path of the coprocessor consists of a 4-by-64-bit register file, exponent control, a 64-bit shifter, a 34-bit arithmetic unit, and a coprocessor PSW. The hardware iteratively performs the various parts of a floating-point operation. For example, a double-precision floating-point addition begins by reading the two operands out of the register file during the EX stage. The exponents of the operands are compared, and the appropriate shift amount is sent to the shifter for denormalization. The denormalized mantissas latch into registers P1 and P2, and the new exponent in register X1. During the M1 pipestage, the arithmetic unit adds the low parts of the two mantissas, and during the M2 stage, adds the carry out and the upper two parts of the mantissas.

The next two instructions convert the respective lower and upper parts of the arithmetic unit's two's-complement result into a signed-magnitude equivalent. The signed-magnitude mantissa is normalized in the next instruction during the M1 pipestage. During the M2 pipestage, the lower part of the mantissa is rounded. The final instruction uses the arithmetic unit during the M2 pipestage to round the upper part of the normalized mantissa, reformat the result into double-precision format, and write it to the register file.

The floating-point coprocessor typically cannot issue an instruction every cycle because of resource conflicts between instructions. These instruction cycles are filled with NO-OPs if no other useful instruction can be rescheduled. These NO-OPs provide the opportunity for the execution of CPU and MMU instructions in parallel with the coprocessor's operation.

Performance. The 200-MHz clock rate of the GaAs microprocessor system is about one-order-of-

Table 1.
CDC 32-bit GaAs microprocessor simulation results.

Benchmark	Gibson mix	Sieve prime number	String search	Linked list
e = Executed instructions	716	7474	892	277
n = Number of NO-OPS	215	2290	256	114
L = Number of loads	65	258	66	32
i = Instruction cache hit rate	93%	99%	99%	96%
o = Operand cache hit rate	91%	96%	95%	94%
R = Rate (1) with simulated cache hit rate; average = 91.0 MIPS	65.2 MIPS	117.4	117.0	67.2
R = Rate (2) with 100% cache hit rate; average = 134.7 MIPS	139.9 MIPS	138.7	142.6	117.7

(1) $R = \dfrac{e \cdot n}{(e + (1 - i)\,ce + (1 - o)\,cL)\,t}$

(2) $R = \dfrac{e \cdot n}{et}$

c = Cache service delay = 16 cycles
t = Cycle time = 5 ns

magnitude faster than in typical silicon microprocessors. However, by factoring out the clock rate from the benchmark-based comparisons, we obtain a more meaningful indication of the architectural efficiency. The primary reason for the GaAs microprocessor system's efficiency is its speed due to the technology and its highly pipelined architecture.

Table 1 shows the net throughput using various benchmarks. The numbers are significantly less than the maximum clock frequency, due to the NO-OPs required in the CPU pipeline to avoid conflicts and due to the limited operand and instruction cache hit ratios.

McDonnell Douglas 32-bit microprocessor. Constraints of the GaAs process drive the architecture and implementation of the 32-bit GaAs microprocessor. As already indicated, project leaders established a strict budget of 30,000 transistors to allow the microprocessor to be fabricated in the near future.

At McDonnell Douglas Astronautics Company, a single-chip microprocessor and a single-chip floating-point coprocessor are under development using a proprietary GaAs JFET process. This project expects to demonstrate developments in GaAs JFET technology and to produce a microprocessor for high-speed, real-time data processing.[20]

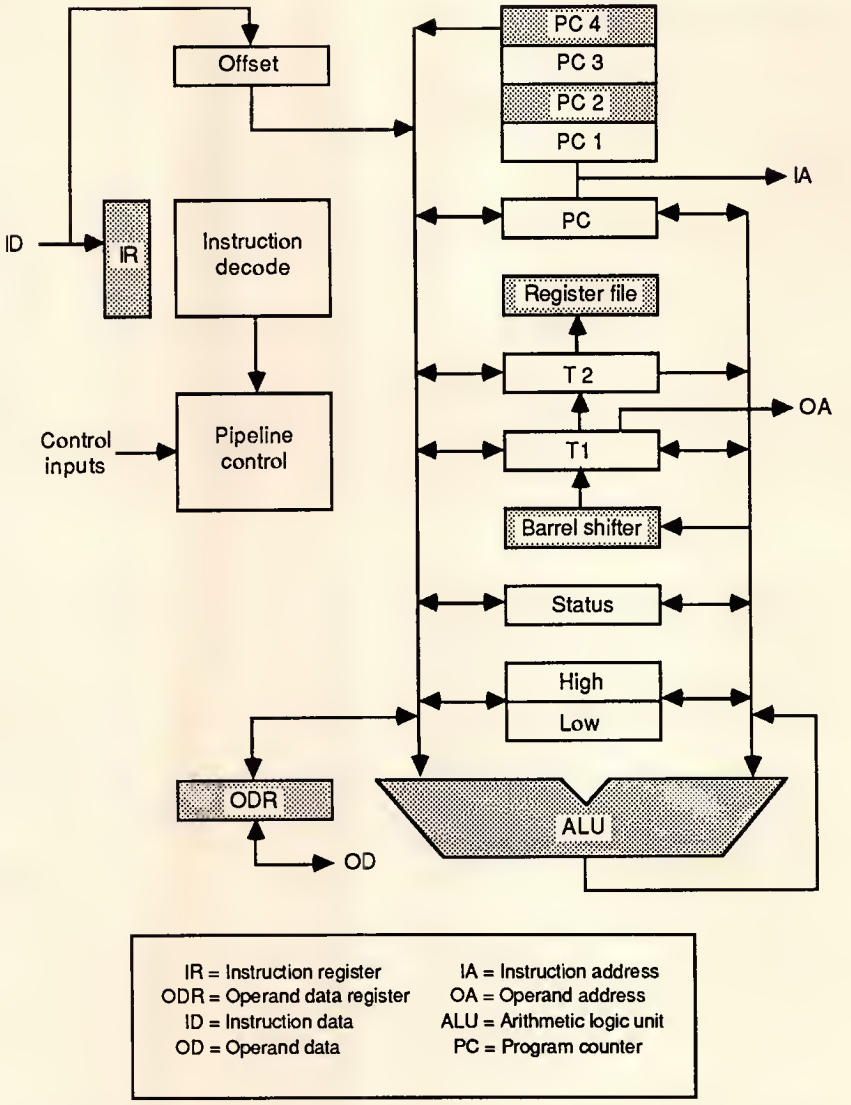


Figure 3. McDonnell Douglas 32-bit CPU block diagram.

CPU architecture. The McDonnell Douglas microprocessor has 17 general-purpose registers and a hardwired "zero generator." A 32-bit barrel shifter shifts, either logically or arithmetically, a 32-bit word by any number of bits in either direction. The operand data register (ODR) communicates with the operand data memory.

All instructions are 32 bits in length and have either three register operands or two register operands and a 16-bit immediate value. Five-bit fields contained in the instructions address program-accessible registers. The fixed format of the instructions leads to a very simple instruction decoding, thus reducing the transistor count required for its implementation and increasing the speed of the system. Instruction decoding and pipeline control circuitry accounts for approximately 5 percent of the total transistor count. The other 95 percent implements the data path.

The multiplication control circuitry is incorporated in the High and Low register pair and the ALU. A 32-bit-by-32-bit multiply processes in 16 cycles and a 64-bit-by-32-bit divide, in 32 cycles. Implementation of a modified Booth's algorithm that operates on a pair of bits at a time accomplishes this feat. Figure 3 shows the CPU data path for this 32-bit microprocessor.

Pipeline design. Designers evaluated two different structures before a choice was made: a four-stage pipeline and a six-stage pipeline with added memory access stages. They determined the better choice to be a four-stage pipeline (we discuss the relative perfor-

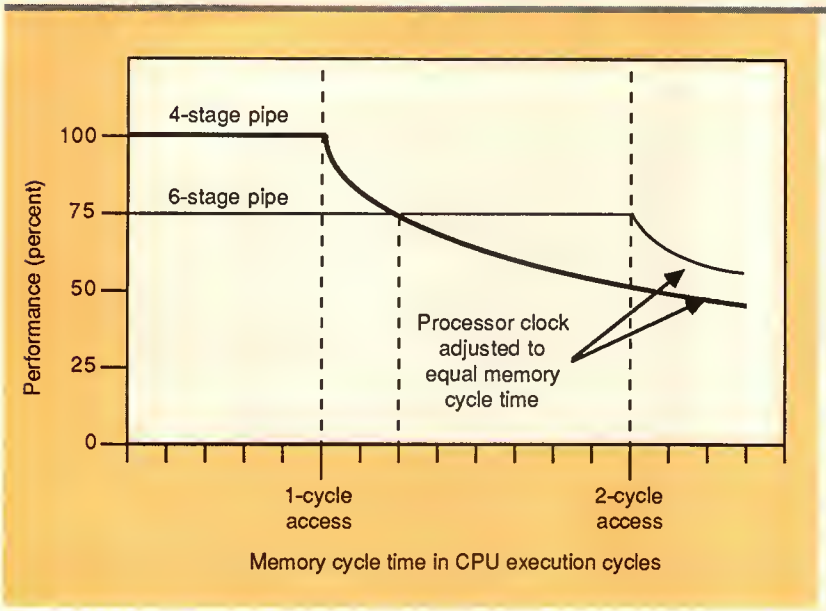


Figure 4. Performances of four-stage and six-stage pipelines.

mances of the two pipelines later). A six-stage pipeline would provide two stages for instruction fetch and two stages for operand read/write. This pipeline would require a pipelined memory system and would increase the branch delay from one cycle to two cycles. In most cases, the code reorganizer would not be able to move useful instructions into both delay slots. Simulations show that with a four-stage pipeline the microprocessor's clock can run at least 25-percent slower than with the six-stage pipeline and still achieve the same net throughput.

The four-stage pipeline contains instruction fetch (IF), file access/ALU or shift operation (ALU), operand fetch or store (OF), and write register back (WB) stages.

An instruction is accessed during the first pipe stage, the IF. If the instruction contains a data value, it is loaded into the offset register concurrently with the instruction register (IR) loading. The operand address fields go directly to the registers so that the operands can be accessed immediately and latched at the ALU and barrel shifter inputs. Due to the simple instruction decode logic, the control lines become valid before the access of the operand is complete.

The result is computed and latched (at the end of the ALU stage) to either the temporary register T1, the PC, the High or Low registers, or the status register (see Figure 3). If the operation is a store, the data is transferred to the ODR during the second half of the ALU stage.

During the third pipestage, the contents of T1 become available on the operand address bus. The data in the T1 can be either the result of an operand computation or an operand address. The operand memory can be either read or written during this pipestage. On a read operation, the operand memory outputs the addressed data onto the operand data bus and makes the data available to the temporary register T2 during the end of the pipestage. If the instruction is neither a read nor a write, T1 simply transfers to T2.

During the last pipestage, the contents of T2 are written to the register file, if this is the destination.

The time it takes for the data to propagate through the critical path of the ALU pipestage determines the microprocessor cycle time. Transferring data between functional units over the internal buses uses 40 percent of the cycle. The rest of the cycle is split between the adder and the register file access.

Cache memory. The high memory bandwidth required by the microprocessor led to separate instruction and operand buses. This decision greatly simplified the cache system design. The separate caches suit the unique characteristics of the individual memories, and each cache size can be smaller without decreasing the hit rate. The instruction cache can be much simpler due to the uniform instruction size and its read-only status.

Floating-point coprocessor. The coprocessor is optimized to perform single- or double-precision floating-point arithmetic conforming to the IEEE floating-point standard.[19] Due to the complexity of floating-point operations, the coprocessor does not follow the single-cycle-instruction execution of the CPU.

Each floating-point instruction represents a complete floating-point operation, including rounding and normalization. The full IEEE standard could not be fully implemented in hardware due to its complexity; therefore the architecture includes only the essential computational elements.

The coprocessor depends on the microprocessor to perform all memory accesses. The CPU monitors the instruction stream, picks out the floating-point instructions as they are fetched by the microprocessor, and executes them. For floating-point load/store instructions, the CPU computes the memory address and initiates the memory access. The coprocessor either loads the data or writes the data onto the operand bus.

To reduce the demands on the microprocessor (related to the fetching of coprocessor instructions), and to allow for greater concurrency between CPU and coprocessor, two floating-point instructions are packed into a 32-bit word.

Multiplication-intensive applications are supported by the existence of multiple floating-point coprocessors in a system. The CPU can support two floating-point coprocessors and two other coprocessors. Because a full multiplier could not fit on the coprocessor chip, designers used a two-bit iterative algorithm. The multiply execution time is about three to four times slower than the add execution time, due to this iterative algorithm.

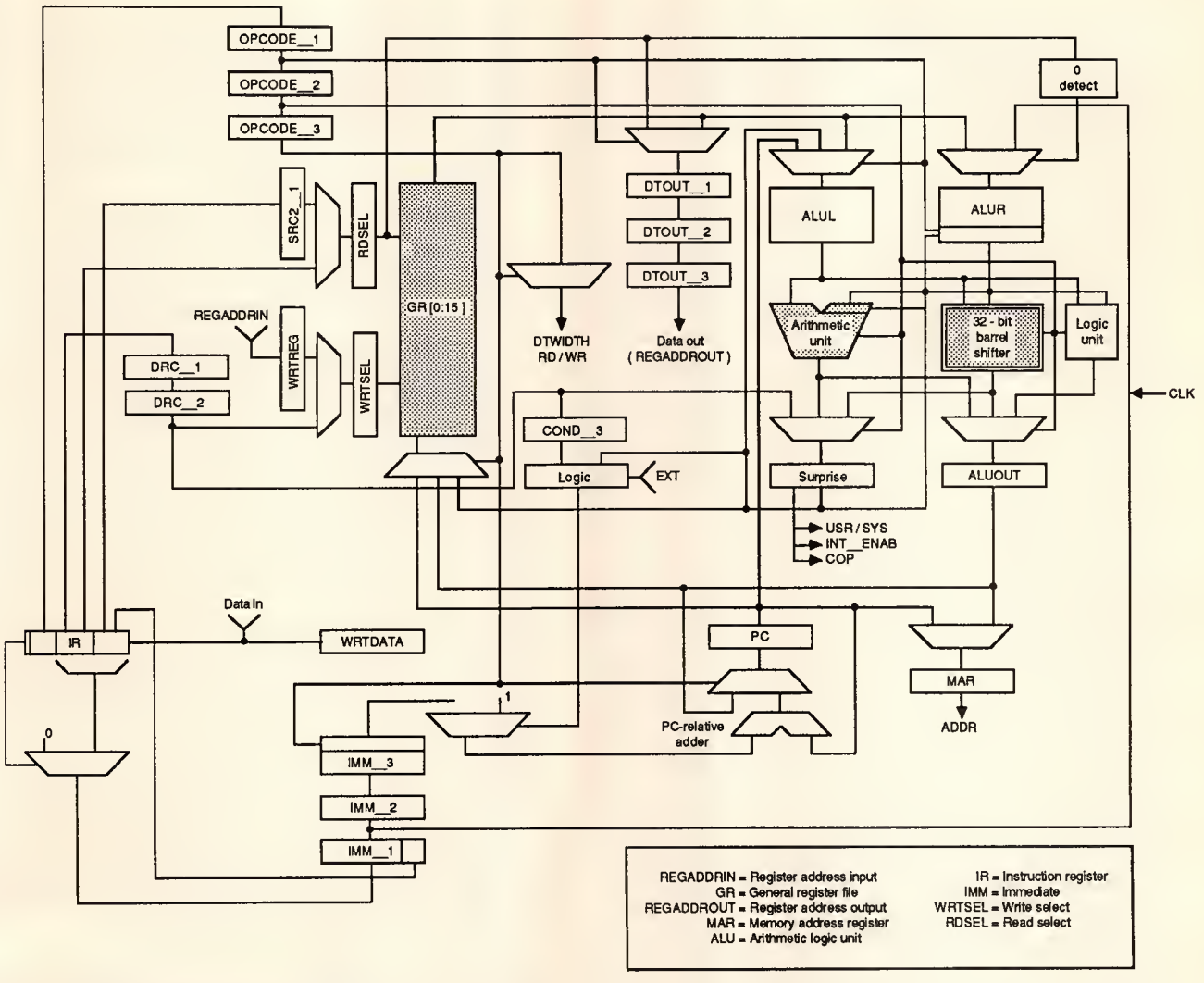


Figure 5. RCA 32-bit CPU block diagram.

Performance. To help the reader, rather than presenting benchmark results, we compare the four-stage and six-stage pipelines to clearly show the rationale behind the choice of the four-stage pipeline. Figure 4 shows the respective normalized performances of a four-stage pipeline and a six-stage pipeline.[20]

RCA 32-bit microprocessor. RCA Corporation also designed a microprocessor patterned along the lines of the Stanford University MIPS microprocessor. The architecture was designed to efficiently execute programs in the conditions typical of GaAs technology. Out of the three microprocessors that DARPA contracted, the RCA design deviated the most from the SU-MIPS baseline.

CPU architecture. Interchip communication in GaAs systems is very costly; therefore, designers must carefully locate the IC on a board to minimize delays. RCA estimated that these delays, per inch of conductor length, take about 5 percent to 10 percent of the clock period. With a CPU clock frequency of 200 MHz, any significant conductor length could cause severe timing problems between ICs. RCA chose to construct a system that will operate independently of the arrangement of the ICs on the board.

The approach used by RCA to overcome the high interchip communication cost sends the destination address of the register in the general register file out from the CPU to the memory, along with the memory address, and returns it with the data as an additional 5-bit field.[21] When the data reaches the CPU, the destination register address field (which arrives with the data) specifies the storage location in the register file. Interchip communication and the memory system can then be pipelined, and no matter how many load requests are active at any given time, everything is kept straight. Figure 5 is a block diagram of RCA's 32-bit GaAs CPU.

With a targeted execution rate of one instruction per clock cycle, RCA has pipelined their system so that up to nine instructions can be in the process of being executed. Consequently, RCA had to concentrate on solving the code optimization problems in deep pipelines.

Pipeline design. The RCA pipeline conditionally consists of nine stages: instruction fetch, wait stage 1, wait stage 2, operation fetch, ALU operation, operation result write back, wait stage 3, wait stage 4, and memory operand write back. The instruction is loaded into the IR during the first stage. The IR connects to multiplexers that sequentially use the general register file address to access operands in the file. The general register file consists of 16 general-purpose registers, one read port, and one write port, but is fast enough so that two pairs of read and write operations can be accomplished in a single CPU clock cycle.

Input logic recognizes whether or not the instruction contains the three most significant bits of the immediate operand of the next instruction. When the next instruction is ready to be placed into the IR, the current contents move to the next instruction register (Next IR). This action occurs simultaneously with the latching of the two operands from the register file into the two registers at the ALU input. Immediate operands are assembled using one byte from the current IR and the three MSBs from the Next IR (if the continuation bit is set). If the continuation bit is not set, only the one byte from the current IR is used to form an immediate operand, with the upper 24 bits set to zero. The 32-bit word is latched as a third input to the ALU.

Now that the three operands are assembled, only two can be selected since the ALU accepts only two inputs. RCA uses an implementation of the register file in which the register "zero" does not exist. If register zero is addressed as the second source operand, the ALU uses the immediate operand. Otherwise, it selects the two register file operands.

For load operations the content of the PC or the Source 1 value (from the register file) becomes the first input to the ALU, and Source 2 or an immediate operand is used for the second input to the ALU. The address is formed and moves to the operand memory to fetch the desired value. A data path for the destination register address also moves to memory to assure correct loading of the data, as described earlier.

For store operations the address is formed just as in the load operation. An additional data path permits the operand from the register file to be supplied to the CPU output. This data path is the same one that is used for load operations.

During the last stage of instruction execution, the output of the ALU moves to its ultimate destination. If the instruction is a memory reference, the address goes to data memory, but the result is not sent back until some time after the next CPU clock cycle starts. If the instruction is an ALU operation, the result is written into the general register file on the first of the two clock phases of this last instruction execution stage. The register file stores all ALU results, including computed addresses. If a result is not needed, it is written to register zero of the general register file and is discarded.

A question that all computer architects must address in the design of pipelined systems is: What happens in a pipelined machine during the cycles following a branch instruction? RCA has added an IGNORE instruction to help reclaim some of the instruction memory space that would be lost to NO-OPs, due to the long branch latency. The code reorganizer of a compiler puts the IGNORE instruction following the last useful instruction migrated into the branch latency area. The count in the IGNORE instruction is set to cause the CPU to ignore the specified number of instructions following IGNORE. The memory space following an IGNORE instruction can be used for the starting instructions of another program module.

Cache memory. Instruction cache is required for high instruction throughput when operating with external memory inaccessible in a single clock period. The cache provides an instruction to the CPU pipeline for each clock cycle on cache hits. If there is a cache miss, the instruction is fetched from external memory and loaded into both the pipeline and the cache. A hit or miss is determined prior to the IF pipestage, with no delay introduced for hit determination. As a result, instruction throughput is not reduced by the presence of the cache.

The cache contains 128 words (instructions) organized into two blocks of 64 words each. The virtual address of the first instruction in each block is stored as the address tag of the block. The remaining locations in the block store up to 63 sequential instructions beginning with a branch target that caused the first cache miss and continuing until the next branch instruction is found. The address tag, which is the starting address for the block, can be any 32-bit address.

A single valid bit is maintained for each instruction location in the cache. Hit or miss determination is made by comparing the Next PC address against the two address tags and checking the valid bits. This cache organization minimizes the amount of overhead logic, and it is designed to capture program loops.

On a cache miss, only a single instruction is fetched from external memory. Block transfers of more than one instruction would improve the hit ratio but would reduce the instruction throughput. Block transfers have a time advantage only if the time required to transfer a block is less than the time required to transfer the instructions separately. Since the hit or miss determination is made early, no penalty other than the instruction transfer time becomes associated with a miss.

If the external memory handles each instruction transfer separately, block transfers have no advantage since they can result in the transfer of instructions that are never used (cache pollution). Also, single transfers allow all externally fetched instructions to be entered directly into the instruction register without first going through the cache.

Floating-point coprocessor. A GaAs floating-point coprocessor accelerates arithmetic calculations in the GaAs microprocessor system. Except for speed, its operation should be indistinguishable to the user from the use of software runtime library arithmetic routines for performing floating-point arithmetic operations.

RCA's GaAs coprocessor converts formats, adds, subtracts, multiplies, and divides in double-precision floating-point format. Single-precision and double-precision integer and floating-point numbers are converted into double-precision floating-point numbers before an operation, and back to their original format when the operation is completed.

The architecture is divided into a mantissa unit and an exponent unit. The mantissa unit is based on an adder/shifter array, which is used to perform adds, shifts by n, multiplies, and divides. The exponent unit is essentially a conventional 11-bit ALU.

This coprocessor contains eight 32-bit data registers, which are configurable as four 64-bit, double-precision floating-point data registers. These memory-mapped registers appear as memory locations to the CPU. Input and output data, to and from the coprocessor, pass through these registers. This scheme is advantageous since the compiler and reorganizer can guarantee that read/write conflicts do not occur between the CPU and coprocessor (in these registers).

Internally, the data register file has two simultaneous 64-bit read ports and one 64-bit write port. Externally, the user addresses the register file as eight 32-bit registers, where the even-address registers hold the least significant half of the double-word values.

The mantissa unit performs all of the arithmetic operations on 53-bit and 23-bit operands. This unit is organized around a multiplier, which is an adder/shifter array capable of performing the multiply, divide (by running the multiplier backwards), mantissa alignment, and normalization operations. Unfortunately, a 53 × 59 adder/shifter array exceeds the limits of current GaAs fabrication technology. To make the architecture practical, RCA reduced the array by introducing some serialism into all of its operations. That is, only a portion of a fully parallel array, operating iteratively, is needed.

Iterative operations allow a slice of the array to perform the function of the full array. Although a very small slice would result in a smaller and less complex chip with lower device count, it would require an excessive number of iterations to perform the arithmetic functions. Each iteration has a small overhead for settling time and propagation of the carries from one slice to the next. Thus, the speed of the array is related to the number of iterations required, which is dependent on the size of the slice.

Performance. RCA gathered the performance evaluation data for the 32-bit microprocessor with an Endot simulation written in the ISP' hardware description language. Most benchmarks that were run on this microprocessor yielded execution speeds below 100 MIPS. For instance, using a bubble-sort benchmark, RCA achieved an execution speed of 76 MIPS.

In GaAs systems the compiler should be considered an integral part of the architecture. With current compiler technology, code optimization in the presence of deep pipelines is not very good. Consequently, the actual execution rate for many benchmarks is well below the peak execution rate of 200 MIPS. When more sophisticated compiler technology becomes available in the near future, this type of GaAs architecture, and GaAs technology in general, can be better exploited.

Other 32-bit efforts. Some R&D efforts directly or indirectly resulted in a 32-bit processor for selected applications. An interesting compiler-writer-oriented view of the problem as well as a number of related ideas can be found in Dietz and Chi[22] and Chi and Dietz.[23] As for the computer-designer-oriented view, Perunicic et al.[24] and Bauer and Meyer[25] present the various theoretical aspects of GaAs arithmetic and GaAs multi-microprocessing resulting from two ongoing PhD projects at Purdue University.

A virtual 32-bit CISC machine in GaAs. The same principles form the basis for processor design in GaAs and silicon. However, the smaller on-chip transistor count and the larger chip-to-chip communication delays, relative to the machine cycle time, yield GaAs designs that differ considerably from silicon designs. Many complex instruction set computer, or CISC, processor designs cannot be directly implemented on a single GaAs chip, due to the inherent differences between GaAs and silicon. This fact renders much of the software written in assembly languages for CISCs useless to the GaAs environment.

One approach in solving this problem is to devote substantial efforts to a multichip reimplementation in GaAs of the CISC host processor. Another, more viable, approach would be to translate the machine-level CISC instructions to equivalent RISC instructions, and to use one of the GaAs architectures previously described for instruction execution.[26]

The advantages of emulating a CISC instruction set on a RISC architecture are numerous. A CISC architecture can be replaced with a GaAs RISC and take advantage of its higher speed and its tolerance to excessive radiation and temperature variations, which

may be useful in some applications. Many application programs in the general-purpose computing area may see improvements in execution speed if a RISC processor is used. Also, by translating CISC programs to the RISC environment, a number of assembly-level application programs become available to RISC architectures. One disadvantage of this approach is that the speed gained is much smaller than the technology speedup. Some overhead is always present in the CISC-to-RISC translation process.

The only restriction to using the translation scheme (to make programs available to both RISC and CISC environments) is that the two architectures must have some basic similarities, such as the same address space and same data bus size. They should also have the same memory map if their I/O devices and coprocessors are memory mapped.

The FACTS Project. The US Air Force has also become interested in GaAs technology and how it may apply to aircraft engine controls. The Wright-Patterson Air Force Base in Dayton, Ohio, funds a research program called Future Advanced Control Technology Study, or FACTS. One of the FACTS program's concerns is technology transferability between very high speed integrated circuits (VHSIC) and GaAs programs.[27]

Allison Gas Turbines Division of General Motors heads the FACTS VHSIC/GaAs program, and Delco System Operations and the University of California at Santa Barbara participate. The program calls for a definition of the current state-of-the-art in GaAs, specification of GaAs requirements for the year 2000, and identification of the technology development required to close identified gaps.

Research topics for GaAs technology include digital and analog signal processing, digital control computing, sensors on chips, and optoelectronic fiber optic interfaces. The FACTS VHSIC/GaAs team focuses on microprocessors, I/O, A/D, D/A, SRAM, and supporting glue chips designed from gate arrays of standard cells.

Microprocessors for symbolic applications

Here, we describe two experimental architectures specifically designed for knowledge-based systems. The recent demand for high-speed knowledge-based systems prompted the design of specialized microprocessors that efficiently execute expert system languages. One such design is the RISCF processor from Carnegie Mellon University. This design incorporated a RISC architecture developed for execution of the OPS5 production system language.[28] The other is the GAELIC processor from Magnavox that was developed for efficient execution of Lisp code.[29]

Carnegie Mellon's RISCF. RISCF is a production system machine architecture implemented in GaAs. Production systems are a special class of expert systems characterized by a set of If-Then rules and a working memory. A production system executes by comparing elements of a problem with the conditional element in the If part of the If-Then rules, and uses some algorithm to choose an appropriate rule among the satisfied rules.

OPS5, the target language for this machine, has several advantages over other production-system languages. In general, it is easier to encode rules in OPS5 than in other types of expert system languages. Unfortunately, OPS5 suffers from the same problems as do all other production system languages—insufficient execution performance.

Optimized RISCF architecture executes the OPS5 language and achieves a significant increase in performance in comparison with general-purpose processors executing OPS5 code.

RISCF architecture. The streamlined instruction set in the RISCF system contains only 32-bit-long instructions. The main data type is the 32-bit integer, and the only arithmetic operations are add and subtract. RISCF's register file is a large structure, 128 words by 32 bits wide, and consists of two read ports and one write port. The register file holds critical elements of the problem space as well as pointers to various matching tests in memory.

A three-stage pipeline in the processor is specifically designed to execute most instruction sequences without pipeline breaks. In developing this pipeline, designers recognized the type of instruction sequences and the frequency of cache misses as important design parameters. The simplicity of the instruction set and the regularity of the operations permit uncomplicated pipelining. An internal forwarding mechanism handles dependencies that may arise between instructions requiring common registers.

RISCF does not require a complex ALU because the predominant operation is that of comparing an operand from memory with a register operand. Consequently, the ALU is simple, having only 10 functions. The ALU can perform one of six operations, each during one machine cycle.

- *Addition*: A simple ripple-carry adder is used.
- *Subtraction*: One of the inputs performs a simple two's-complement conversion, and then the addition function performs the subtraction.
- *Signextend*: The first 19 bits pass directly to the output. The 19th bit is copied to the remaining bits of the 32-bit word and then passes to the output.
- *Hiload*: The value of the lower 16 bits of input 1 passes to the upper 16 bits of input 2, and the lower 16 bits of input 1 remain unchanged.
- *Passleft, passright*: The left or right input to the ALU passes to its output.
- *OR, AND, NOT, XOR*: These functions are simply implemented using NOT and NOR gates.

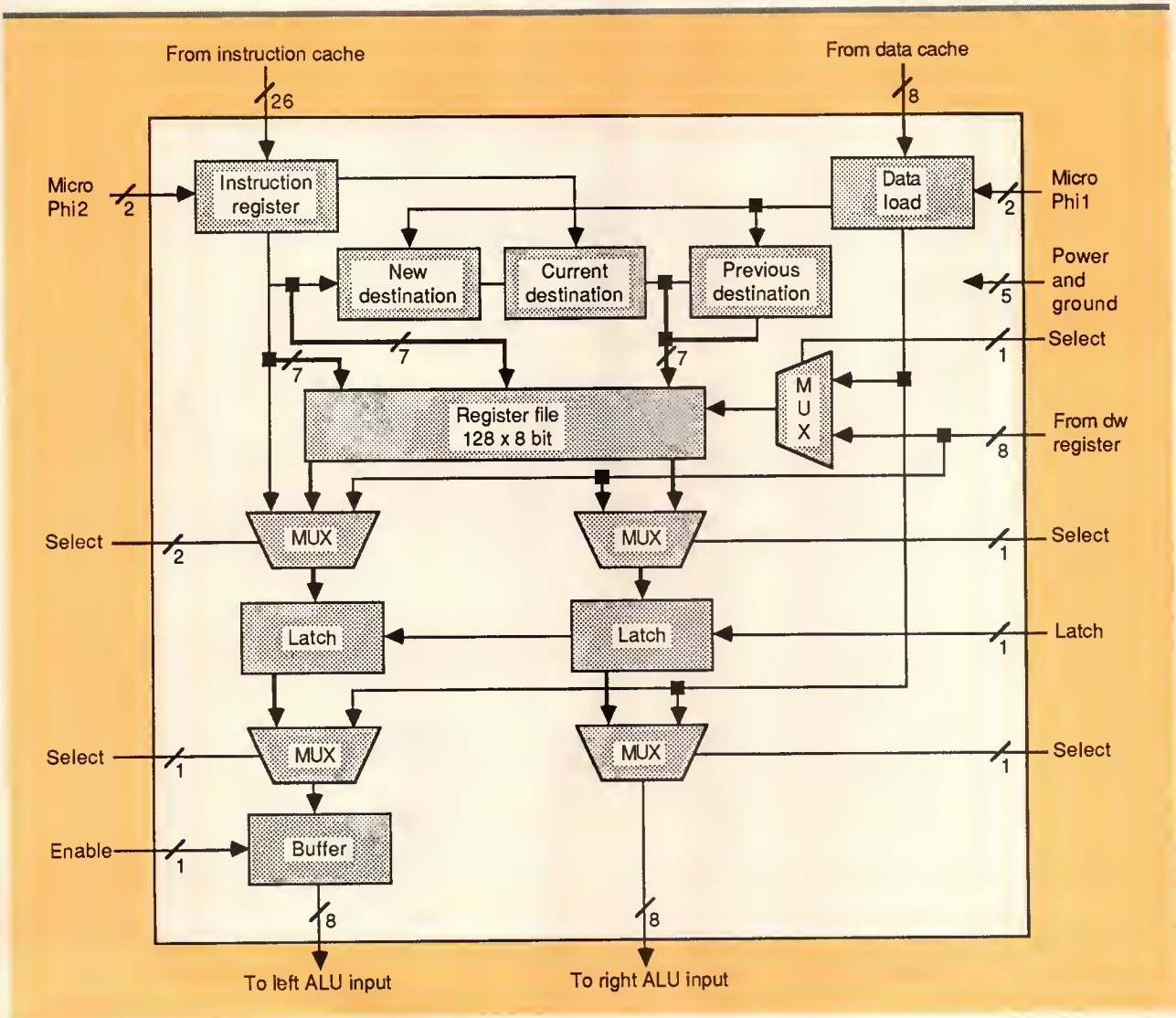


Figure 6. RISCF register file module.

For a RISC processor to be realized in GaAs, it must have a high-memory bandwidth to achieve acceptable performance levels. RISCF has dual caches, one for instructions and one for data, which permit simultaneous access to both instructions and operands. Studies have shown that a data cache 32K bytes by 32 bits can hold over 90 percent of all data references.[28] However, a much smaller cache would probably be sufficient because OPS5 data references frequently exhibit a high degree of locality.

Modularization of RISCF. Since RISCF was intended to be fabricated using D-MESFET technology, it had to be decomposed into modularized chips to be integrated into the LSI levels currently attainable in D-MESFET systems. The advantage RISCF might obtain from a GaAs technology that has not matured to VLSI levels depends on how well it can be modularized to reduce the effects of interchip delays.

Knowledge of the approximate number of gates needed for each substructure, the shared communication paths, and the maximum pin-out allowed for a GaAs chip is required before the processor can be modularized. A good scheme to decompose RISCF would be to keep critical paths and closely related substructures within individual modules. After careful evaluation, Carnegie Mellon identified these modules as the register file, the ALU, the instruction fetch, and the data fetch. Each of these modules contains substructures like registers, incrementers, decrementers, multiplexers, adders, and the ALU. In assigning the gate geometry to these substructures, designers used only NOR and NOT gates and paid no attention to the different transistor configurations used to implement these gates in GaAs.

These structures were used to build each of the four modules of RISCF. Keeping communication distances short is an important layout concern because of the relatively long interchip delays.

Critical paths. The critical path in the register file module (Figure 6) begins at the address inputs to the file and ends at the pre-ALU latches. This delay

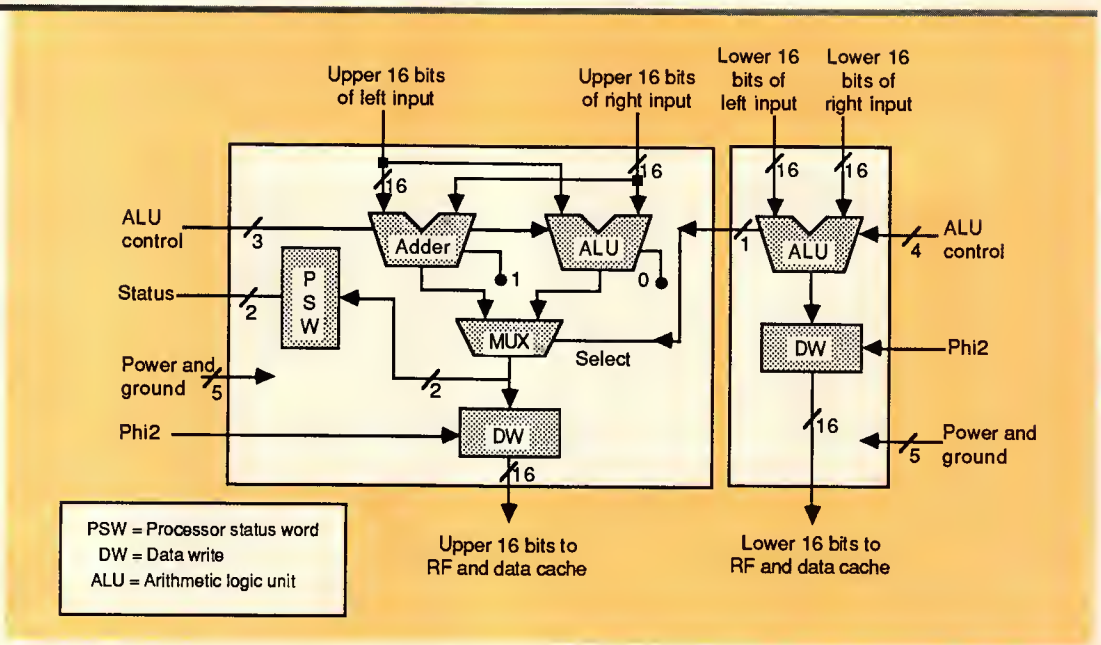


Figure 7. RISCF instruction fetch module.

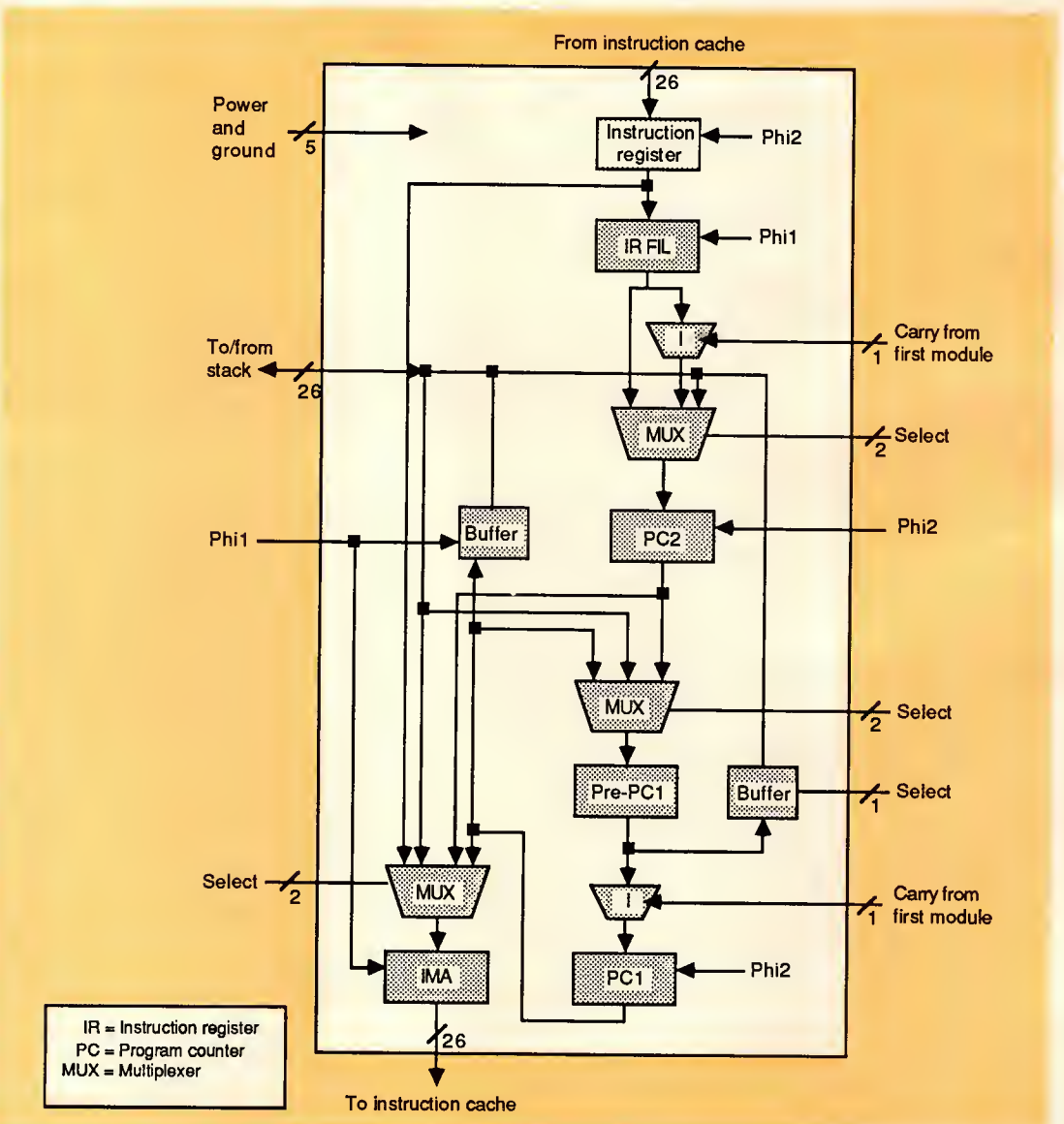


Figure 8. RISCF ALU module.

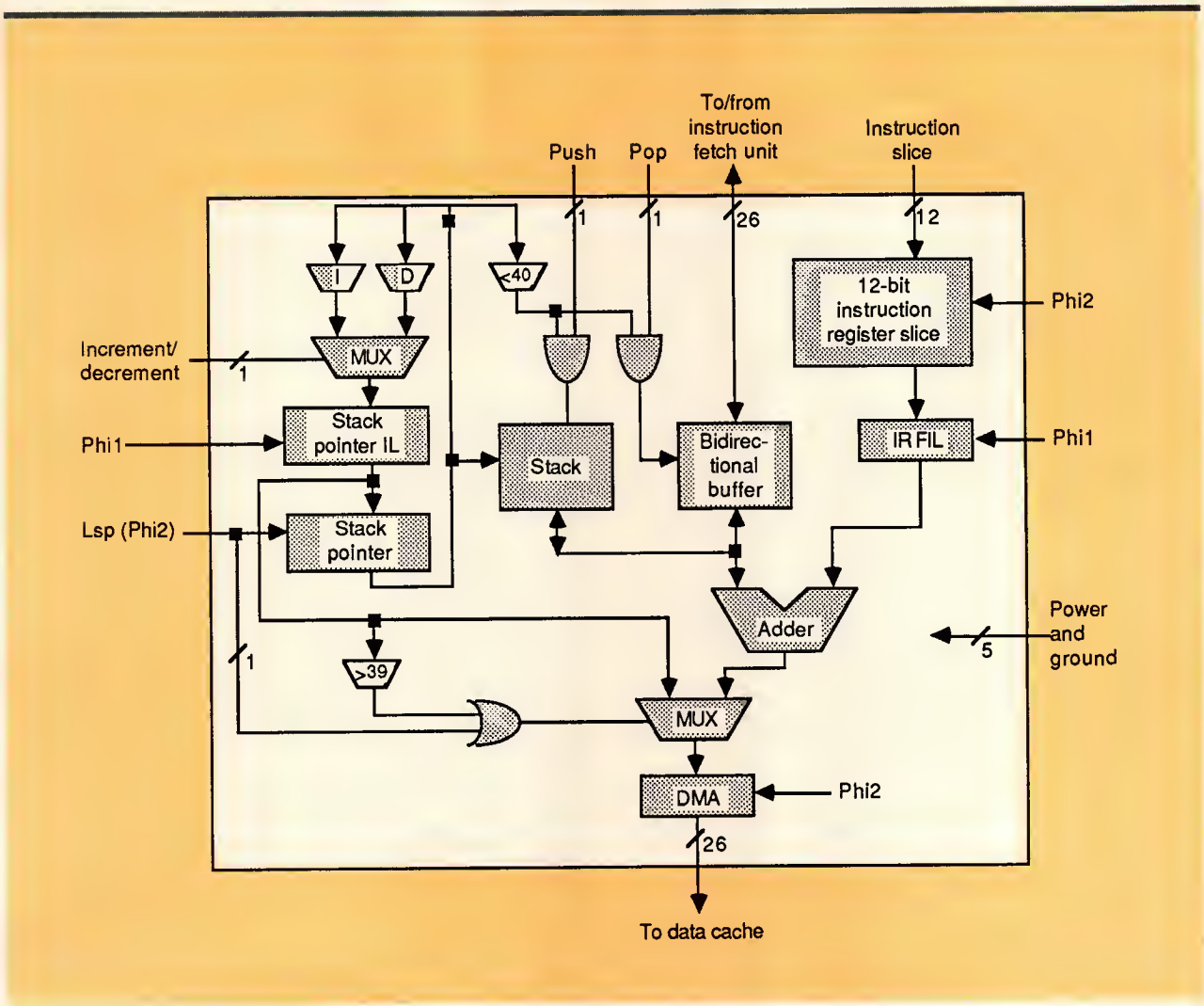


Figure 9. RISCF data fetch module.

occurs during the $\phi 2$ phase of the clock. The delay through this path is:

4500[RF] + 3500[MUX] + 3 * 1000[MUX drivers] + 320[Latch] = 11320 ps.

The critical path through the ALU module (Figure 7) occurs during the $\phi 2$ phase of the clock. The carry from the lower 16-bit module passes to the other ALU module multiplexing the appropriate result from the two 16-bit ALUs. The module delay is:

12500[ALU + Driver] + 320[Registers] = 12820 ps.

The critical path in the instruction fetch modules (Figure 8) occurs during the $\phi 2$ phase of the clock and begins at the field isolation latch (IRFIL). It extends through the incrementer and target address multiplexer (TAM) until it is latched in the secondary program counter. The delay is:

7540[INCR] + 1000[Driver] + (2860 + 320)[MUX] + 320[PC2] = 12040 ps.

The adder used to compute the fetch addresses latches the delays in the data fetch module (Figure 9). The critical path begins at the stack output, extends through the adder and data address multiplexer (DAM), and ends at the data memory address register (DMA). The delay occurs on $\phi 2$ and is:

11830[26-bit adder] + (2860 + 320)[MUX] + 320[DMA] = 15330 ps.

The longest intramodule delay for $\phi 1$ is about 11 nanoseconds, and for $\phi 2$ it is about 15 ns. A more accurate measure of $\phi 1$ and $\phi 2$ is 13 ns and 17 ns, respectively, when the drivers are taken into account. This translates to a machine cycle time of roughly 30 ns.

RISCF comments. Although GaAs devices promise high speed, they have substantial problems in fabrication and implementation. When GaAs technology reaches VLSI levels, all these modules could be combined onto a single chip, thus eliminating interchip communication delays. Carnegie Mellon expects the current design of the RISCF modularized processor, with a 30-ns cycle time, to execute OPS5 code 20 to 40 times faster than a VAX 11/780.

GAELIC from Magnavox. Highly parallel processors have shown enormous potential in symbolic computing over the past few years. Cost-effective, powerful knowledge-based systems have been pursued by a variety of means, including multiprocessors, new and innovative architectures, and materials with higher electron mobility.

The Gallium Arsenide Experimental Lisp Integrated Circuit, or GAELIC, is a high-speed processing element of a GaAs multiprocessor system optimized for the execution of Lisp code.[29]

Design objectives. The principal design objectives of GAELIC were to maximize execution speed of symbolic programs and to minimize the transistor count so that the design would fit into GaAs. A RISC architecture is necessary to satisfy both goals.

The instruction set of GAELIC provides direct support for many Lisp primitives. These primitives are the instructions that occur most frequently in Lisp programs. Lisp programmers usually program in complex functions, which can be decomposed into the underlying Lisp primitives by a compiler and optimized for execution on GAELIC.

GAELIC architecture. The compiler's view of the GAELIC processing element consists of a four-billion-entry symbol table, one million addressable lists (where each list may have up to 4096 nodes), and an "infinite" stack.

Four address spaces are associated with GAELIC: the instruction space, the list space, the stack, and the symbol table. Six supporting units are required to manage these four address spaces:

- The *instruction manager* keeps track of the program counters, and maintains an instruction prefetch buffer and a history of old PCs. It also executes directives from the compiler.
- The *instruction cache* backs up the instruction manager.
- The *garbage collector*, a separate processor, collects and compacts the list space in parallel with GAELIC processing.
- The *list space* works as memory for lists, where a portion of the address space extends onto the GAELIC chip.
- The *auxiliary memory* works as memory for the stack, whenever the top of the stack extends onto the GAELIC chip.
- The *symbol space* stores atoms and their attributes. A portion of the symbol table may reside on the GAELIC chip in a part of the region normally occupied by the list space.

The user sees GAELIC as a pure Lisp machine that supports a wide variety of Lisp constructs. Since Lisp makes no distinction between hardware-supported functions and those defined by a user (or in a runtime library), "large Lisps" such as CommonLisp, ZetaLisp, and InterLisp can run efficiently on GAELIC. Many programs in these languages make extensive use of integer and floating-point arithmetic, and array data structures. Because GAELIC is optimized for symbolic processing, it would be more suitable to run such programs on a multiprocessor system that includes both symbolic and numeric processors.

Micro-ALU. The micro-ALU can best be defined in terms of what it does not support. GAELIC does not support any form of bit or byte manipulation, nor is there any user-accessible form of shift. There is no hardware support for floating-point numbers, arrays, strings, or records. There is no addition, subtraction, or multiplication (except for Peano arithmetic: increment, decrement). There is no logical or bit-wise AND, OR, XOR, and NOT. Also, there are no equality or inequality operators. The EQ predicate primarily compares atoms to see if they are the same. GAELIC has no program counter and thus no branch instructions in the traditional sense.

All ALU instructions are performed relative to the stack. Items are pushed onto the stack using the PUSH instruction from either the symbol table (atoms) or list space. Items are popped as they are used by instructions. There are no data or address registers.

Lists. GAELIC uses a compact list representation developed at the University of Illinois at Urbana. This representation is based on the fact that Lisp lists are, with a few exceptions, simple binary trees. These exceptions are stored in an exception table. Exceptions occur for only five reasons:

- a nonnumeric atom,
- a numeric atom,
- an end-of-list find,
- a reference counter greater than one, and
- a forwarding pointer pointing to another exception table.

Nodes on the binary tree are numbered by three rules:

- The base node of an exception table is numbered 1.
- The CAR function of node n is at $2n$ unless node $2n$ has an exception.
- The CDR function of node n is at node $2n + 1$ unless node $2n + 1$ has an exception.

To find the CAR of a node, we use the following recursive algorithm:

1) Present (CAR n) = $2n$ to the list file.
2) If no exception is found, the CAR is at node $2n$. Push $2n$ onto the stack.
3) If a "multiple reference count" exception is

found, ignore it. CAR is at $2n$.

4) If either of the atom exceptions is found, look up the name of the atom in the exception table and return it onto the stack.

5) If the end-of-list has been found, return NIL onto the stack.

6) If a forwarding pointer is found, present the forwarding pointer to the list file and go to Step 2.

GAELIC microarchitecture. The four stages of the GAELIC pipelined architecture are instruction fetch (IF), instruction decode (ID), execute (EX), and operand write (WR). These stages were chosen as a compromise between extra transistors and the speedup available from increased pipelining.

GAELIC uses a hardwired instruction decoder, as do most RISCs, rather than a microprogram. This approach conserves space on the chip and is made feasible by the streamlined instruction set. Opcodes for GAELIC fit within a single byte. Since nearly all instructions are stack oriented, few instructions require more than a single byte to encode. Thus four instructions can be packed into a single 32-bit word, quadrupling the effective instruction bus bandwidth.

Two instruction registers (IR1 and IR2) supported on chip make a total of eight instructions available at any given time. The two registers become useful on conditional branches in which instructions that are normally taken from one register can also be taken from the second register, which holds instructions from the branch target. The inactive instruction register is loaded onto an idle bus cycle prior to the branch. If the branch is invoked and the inactive register has not been filled, the processor purges itself and asks the instruction manager for the other address.

When the instruction stream switches to the inactive IR (say IR2), the instruction manager looks for the next PRELOAD instruction and loads IR1 with instructions from the next branch target on the next free bus cycle. This approach significantly decreases the number of wait states spent on conditional branches. In addition, the optimizer can sometimes fill those wait states with useful instructions.

Since GAELIC fetches instructions every fourth machine cycle, when a branch is taken there are at most five instructions on chip (four in the IR and one being decoded). The processor continues to execute these until the instruction manager asserts the control signal PURGE. This allows the optimizer to attempt to fill the slots following a branch with instructions that do not depend on the stack, such as unbinding atoms that are no longer needed.

During the instruction decode stage, the hardware identifies the instruction and the presence of any operands. If they are not yet present, they are brought into the on-chip associative memory. This stage also manages the stack, symbol table, property lists, and exception tables.

GAELIC executed Lisp code at 300-1000 MIPS, speeds well beyond any attained by commercially available symbolic processors.

The execute stage is the most complex stage in the pipeline, since instructions may iterate before completing. See the recursive definition of CAR described earlier for an example.

During the operand write stage, the instruction manager and the processor take care of any stack, symbol, or list space evacuations. A PRELOAD initiated in this stage can usually be ready by the next execute stage.

Performance. Simulations suggest that GAELIC can execute at speeds approaching 1000 MIPS when clocked at 1 GHz and with all wait states eliminated. By including the wait states, the effective speed decreases by a factor of 2 to 5, depending on the efficiency of the code optimizer.[30] If the programs executed are typical AI programs, such as a backward-chaining inference engine, 300 MIPS can be expected. This execution speed equates to about 300K to 3M logical inferences per second (LIPS). When used as part of a 128-processor element, multiple-instruction, multiple-data (MIMD) hypercube network, speeds on the order of 15 billion LIPS are possible.[29]

Comments. While GaAs makes significant speed increases possible, it does so at the cost of a small transistor count and relatively long off-chip accesses. GAELIC solves the first problem by using a RISC architecture and dispensing with many of the features normally found in a processor. The second problem is solved by using an on-chip associative list file, using several on-chip top-of-stack registers, and packing multiple instructions into a single 32-bit word to increase the instruction bus bandwidth.

GAELIC was an experimental design for symbolic processing capable of executing Lisp code at speeds of 300 MIPS to 1000 MIPS, speeds well beyond any attained by commercially available symbolic processors.

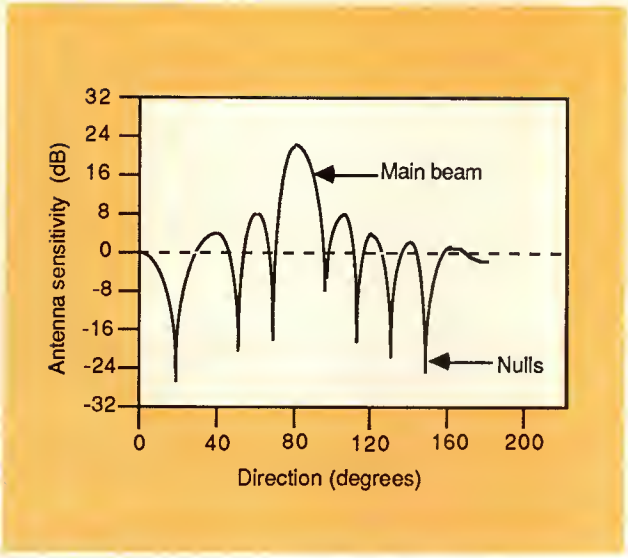


Figure 10. Typical radar antenna array pattern.

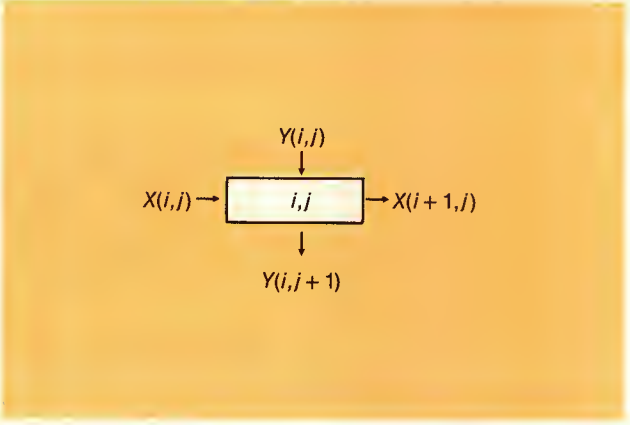


Figure 11. Algorithm mapping.

Systolic array processors

The increasing computational power required of many signal processing applications is exceeding the capacity of uniprocessor systems. Multiprocessor systems offer a cost-effective means of increasing computational power. In many real-time signal processing applications, high-speed processing hardware is required. Some of the advantages of using a GaAs processor for these applications are its high speed characteristics, its radiation hardness, and its tolerance to temperature variations.

RCA systolic array processor. Systolic array architectures can be used in applications in which intense matrix arithmetic or highly iterative algorithms need to be implemented. One such application is digital radar beamforming. Here the systolic array is an array of processors, each communicating with its nearest neighbors and configured as a single-instruction, multiple-data (SIMD) machine. The processors are simple in design, because they are custom built to perform only one algorithm. RCA designed a systolic array that will be used in an application of digital radar beamforming with automatic null steering.

System application. The system performs in radar applications in which jamming signals strike a radar antenna from many different directions. A typical radar antenna array pattern consists of a main beam and multiple sidelobes and nulls as shown in Figure 10. To attenuate the jamming signal, a beam controller generates a set of coefficients that produce an antenna pattern in which the nulls are in the directions of the unwanted signals. When the jammer changes position, the controller moves the nulls to continuously maximize the attenuation of the jamming signal. At the same time, the controller must maintain a highly directive main beam in the desired direction.[31]

The computation of the coefficients involves matrix arithmetic that can be performed more rapidly by a systolic array architecture than by conventional architectures.

Algorithm mapping. The algorithm that would adaptively generate the complex coefficient vector must have a combination of performance, stability, and ease of systolic implementation. Performance is judged by the speed of convergence, the depth of nulls, and the shape of the antenna pattern produced. Based on these criteria RCA selected algorithms MSR3, Gram-Schmidt, and Givens for evaluation.

The MSR3 algorithm served as a guide for designing the systolic array.[32] The computational core of the algorithm consisted of a pair of doubly nested Do loops with several arithmetic operations within the loop. For a doubly nested loop, a two-dimensional array of processing nodes could be used. Each array node corresponds to an iteration of the outer loop, and each column corresponds to an iteration of the inner loop. The upper limit of the inner loop depends on the index of the outer loop. For example:

```
FOR i := 1 to N DO
  FOR j := 1 to i DO
    X(i + 1 , j) := X(i , j) * Y(i , j)
    Y(i , j + 1) := X(i , j) * Y(i , j)
```

This results in a triangular array of nodes in which each node is defined as shown in Figure 11. The

arithmetic in the computational core of the algorithm consists of multiply and add operations with one real divide in the outer loop.

RCA ran simulations to verify the direction of data flow in the systolic array. In the first pair of Do loops that compose the algorithm, data flows from right to left and top to bottom, while in the second pair of Do loops data flows from left to right and bottom to top. A problem occurs in pipelining these operations when the algorithm data flow propagates in two different directions simultaneously. This condition leaves most of the nodes idle the majority of the time.

The chosen system architecture achieves the proper algorithmic data flow by providing storage locations at each node and allowing both waves to run simultaneously in opposite directions on the triangular array of processors. These storage locations hold critical values from each stage of a pipeline of operations in a nodal processor. The depth of the pipeline is equal to twice the number of processors along the diagonal of the array. Under steady-state processing, the processors can be made to run the single sequence of operations in three phases, with 100-percent utilization of the interior processors. The most reliable way to achieve the steady-state condition is to initialize the state of the array to the steady-state condition on start-up.

CPU architecture. The system architecture set certain constraints on the internal architecture of the nodal processor. It required communication I/O ports on each of the four sides of the nodal processor. RCA serialized the data ports by nibbles to lessen the total complexity of data ports. The register set of the processor consists of four general registers that are used for holding critical values of the algorithm between operations.

Designers determined the clock rate of the processor to be 80 MHz and the data transfer rate between processors to be 12 MHz. The system would update the coefficients for the multiple beams every 5 μs to meet the specifications of the radar system update time for coefficient vectors. The data is transmitted serially over eight lines, two unidirectional lines on each of four sides of a processing node in the array. Two handshake lines, data valid and acknowledge, exist for each data line.

The designers optimized the ALU to perform the functions needed by the algorithm. The functions were implemented with the most efficient hardware circuits to meet the low gate-count restrictions of GaAs technology. The basic architecture of the ALU is shown in Figure 12 on the next page.

The ALU has two 10-bit inputs and one 10-bit output. The 10-bit package includes a 2-bit exponent, a sign bit, and a 7-bit mantissa. These 10 bits represent a pseudo floating-point number with the equivalent range of a 14-bit, fixed-point representation. The pseudo floating-point format was developed to meet the range and accuracy requirements of the beamforming application.

The exponent can put the value of the mantissa into one of the ranges listed in Table 2 on the next page. Each range has a value four times greater than the previous range. An unnormalized mantissa eases the interface to the outside world of fixed-point representations.

Two 10-bit registers and two 7-bit registers make up the ALU. One of the 7-bit registers is a simple latch set; the other three are master-slave flip-flops. The ALU also has a 7-bit combination ripple-carry/carry-save adder, a 7-bit shifter, exponent and sign-handling circuits, and the necessary multiplexers. A left-by-one, left-by-two, right-by-one, right-by-two shifter is implemented with minimal hardware, but requires extra steps to perform longer shifts.

The ALU architecture was designed around a special 7-bit adder circuit that has the capability of operating in either carry-save or ripple-carry mode. Carry-save provides a very fast multiply operation, whereas ripple-carry is needed to propagate the carries from the final stage of multiplication and also for normal add and subtract. The adder uses two operands: operand register A, and either operand register B or the result of the adder (known as a partial product). Operand A requires shift left-by-one capability, and the partial product requires shift right-by-one capability. This capability is needed only during multiplication.

To produce a set of coefficients every 5 μs, we must perform a real multiply on the order of every 150 ns. Logic simulations indicate that the ALU can perform a real multiply in less than 45 ns, giving a threefold safety margin.

The reciprocator hardware had to be kept to a minimum due to the constraints on gate count. Therefore, a minimum-hardware implementation was used. It consisted of an 8-value lookup table for the initial guess (implemented in combinational logic) and an additional multiplexer in the ALU. The hardware required for normalization/denormalization of the reciprocator, along with mantissa adjustment needed to maintain the pseudo floating-point notation, consists of a shifter and some control logic with the ALU registers used to temporarily store the intermediate values. Since reciprocation only occurs in the main-diagonal nodes that have less computational load than the other nodes in the array, the reciprocation process does not cause any timing problems or bottlenecks. Since a multiplier and two's complementer already exist in the ALU, RCA chose a convergence algorithm using only multiplication and two's complementing to approximate reciprocation.

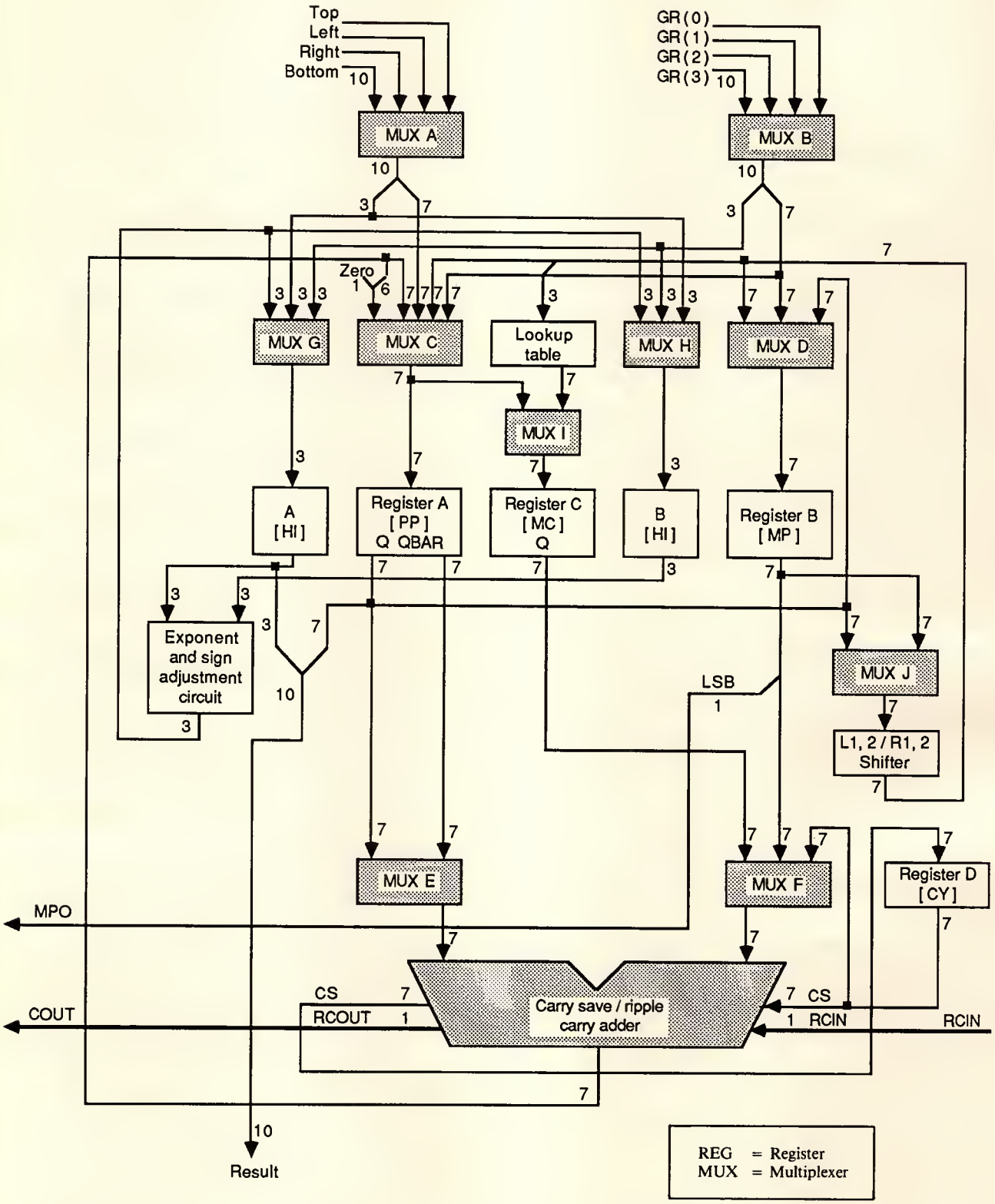


Figure 12. Systolic array processing node architecture.

Comments. In systolic array architectures, choosing a sophisticated algorithm to complement the intended application is as important as the hardware efficiency of the overall design. Since the nodal processors in a systolic array are custom built to execute a single algorithm, an optimal match between application and algorithm is necessary to obtain a good design.

After an efficient algorithm has been selected, the characteristics of the algorithm must be examined and mapped to an architecture. Designers then evaluate and optimize the architecture for the conditions of the underlying chip fabrication technology. In the case of GaAs, the nodal processors have chip gate-count limitations as well as other conditions that must be taken into account to obtain an efficient architecture with a high throughput, fast response time, and hardware efficiency.

Alternative approaches to processor design

Two alternative approaches to processor design in GaAs are wafer-scale implementation (WSI) of a RISC-style processor, from the Rensselaer Polytechnic Institute,[33] and a microcontroller from Purdue University, which is also being considered for WSI implementation.

The motivation behind choosing a WSI architecture is the decrease of interchip communication costs, which are very significant in GaAs systems. The wiring between chips on a wafer is much shorter than it would be if each chip was on a separate package. The chips on the wafer are connected with very dense wiring, which is routed around chips that are nonoperational due to die defects.

WSI of a GaAs architecture. Recent improvements in the processing of raw GaAs wafers have encouraged computer architects to examine the impacts of wafer scale implementations on large systems. WSI and wafer scale hybrid packaging (WSHP) offer the possibility of assembling large systems from small building blocks on a single wafer that can be fabricated at a reasonable yield.

The approach used to design a RISC processor on a wafer is a particular type of WSHP called a wafer transmission module (WTM). The wires on the wafer are fabricated thick enough to act as LC transmission lines that may be run over the diameter of the wafer. The line delays are dictated by the speed of the electromagnetic wave propagation in the multilayer dielectric when the line is terminated properly. Special circuits are required to drive these microtransmission lines. The speed achieved using the WTM approach over that achievable in a single-chip VLSI processor is limited by the delays associated to the line drivers and by the loss in circuit density resulting from the insertion of bond pads and extra die spacing for wire routing.

Table 2.
Psuedo floating-point number ranges.

Range		Mantissa		Exponent
Lower	Upper	Lower	Upper	
0	±255	0000000	1111111	00
±256	±1023	0100000	1111111	01
±1024	±4095	0100000	1111111	10
±4096	±8192	0100000	1111111	11

The WTM implementation of a RISC processor described here is a design from Rensselaer Polytechnic Institute that uses building blocks of 200-gate circuit density. RPI expects future designs to employ 1000 or more gate circuits per die.

CPU architecture. The architecture considered for a WSHP implementation of a 32-bit RISC in GaAs is shown in Figure 13 on the next page. The data path has two internal buses, the L-bus and the S-bus, and two external buses, the instruction bus and the data bus. The Source1 operand transfers directly on the L-bus to the ALU input latch A. The Source2 operand transfers to the ALU input latch B on the S-bus after passing through a shifter. The shifter performs one of four operations: pass, shift left-by-one, shift left-by-two, and shift right-by-one.

The register file contains 16 registers with two read ports and one write port. This organization allows the ALU to access two operands from the register file and concurrently perform a write to a register file, all within a single system cycle. The result of an ALU operation transfers through the L-bus to the feed-forward register and writes to the register file on the next cycle. This delayed-write scheme reduces the system cycle time by eliminating the relatively long register-file write delay from the critical path. If the destination register is equal to one of the source operands of the following instruction, the operand is taken from the feed-forward register (instead of the register file that has not yet been updated). This process eliminates destination-source conflicts that might arise due to the delayed write scheme.

The ALU only performs four functions: ADD, AND, OR, and XOR. Subtraction is implemented by inverting one of the operands and the carry-in (to generate a "borrow-in") followed by an ADD operation.

The program counter logic includes four registers, an incrementer, and an I/O multiplexer. The NEXT_PC is normally loaded from the output of the incrementer, but in the case of a BRANCH

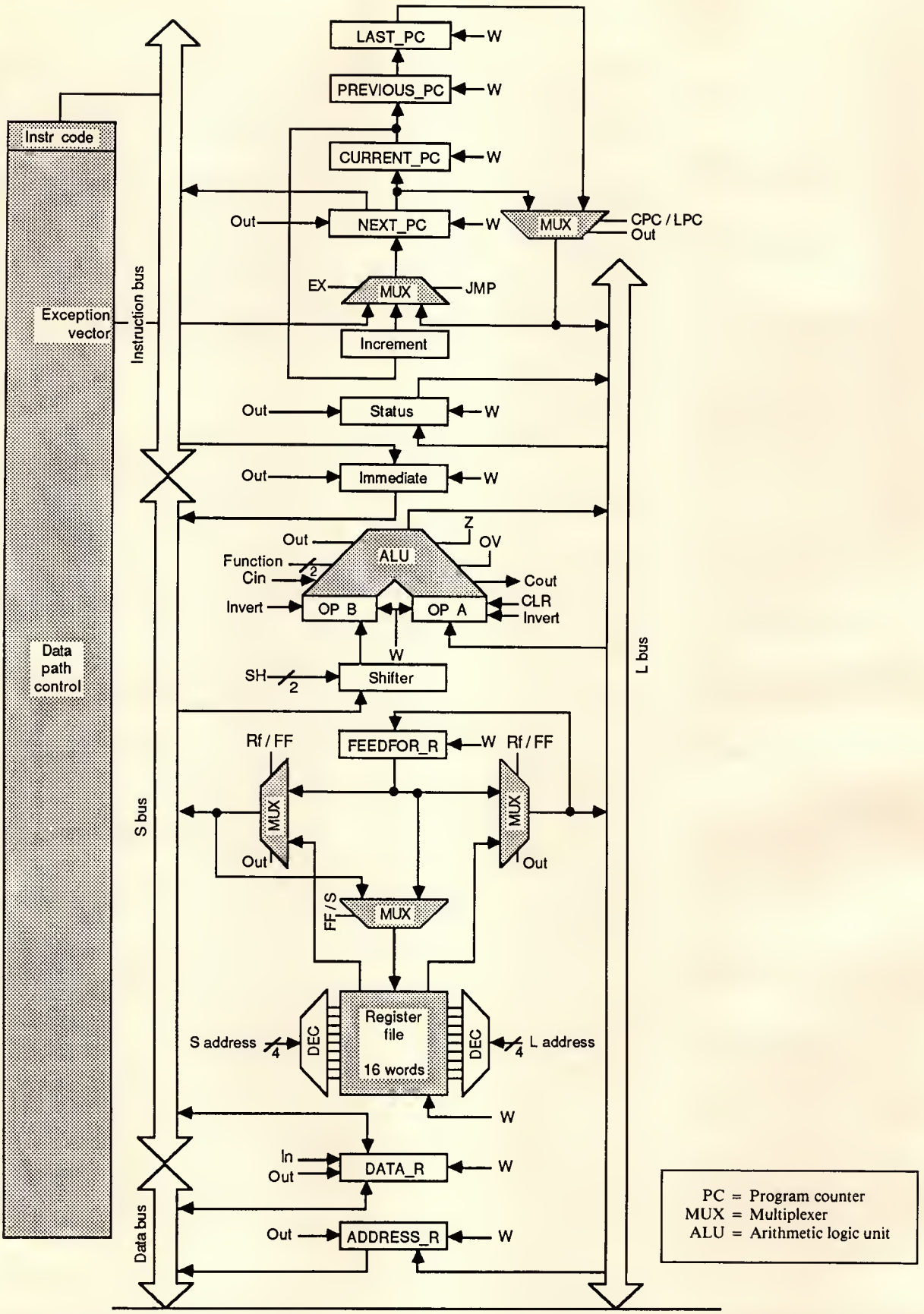


Figure 13. Wafer scale integration of a RISC processor.

instruction it can be loaded with a jump address from the ALU. In the case of an interrupt, reset, or I/O trap, it is loaded with the corresponding exception vector. The value of the CURRENT_PC transfers through the L-bus to the ALU target for the jump-address calculation. The LAST_PC can also transfer to the L-bus to save the return address after an interrupt or a trap. The PREVIOUS_PC and LAST_PC are necessary for reloading the pipeline with the instructions that were executing when the interrupt or I/O trap occurred.

The pipelined architecture (combined with the separate instruction and data buses) allows ALU operations, instruction fetch, and data I/O to execute in parallel. The three stages of the pipeline are instruction fetch, execute cycle, and data I/O cycle. The delays of the critical path define the cycle time of the processor. The critical path consists of the instruction decoder, address decoder, register-file read, shifter, ALU latch, ALU (ADD operation), and finally register-file write.

Instruction set. The instruction set for this RISC processor consists of 15 instructions, all fixed to 32 bits in length. The number of instructions is not an upper limit of what can be implemented.

The format of most instructions is Operation <Source1>, <Source2>, <Destination>. The Source2 operand of the instructions can be replaced with a 12-bit immediate constant, which can be passed through the shifter before being latched as an ALU input. Branch instructions are given as BRANCH <Imm. Constant>, <Condition> and load instructions, as LOAD <Imm. Constant>, <Destination>. For the BRANCH and LOAD instructions, the immediate constant can be 20 bits wide and can be complemented. I/O addresses can be formed by adding Source1 and Source2 operands. There are no subroutine CALL and RETURN instructions. These must be constructed from the available instructions. One instruction that is particularly useful is the GETLASTPC instruction, which is used to save the return address after an I/O trap or interrupt has occurred.

Performance. RPI estimates the system cycle time to be 15.8 ns using information obtained from TriQuint Semiconductor. This translates to an execution rate of 63 MIPS. However, delayed branch schemes and source-destination conflicts after load instructions will decrease the execution rate by a few percent. To achieve a nearly maximum throughput, the memory system must be able to supply one instruction and one data word to the CPU every cycle. A memory bandwidth of 506M bytes/s would be required if a single memory is used for both instructions and data. An alternative approach would be to utilize fast data and instruction caches and thereby reduce the main memory bandwidth.

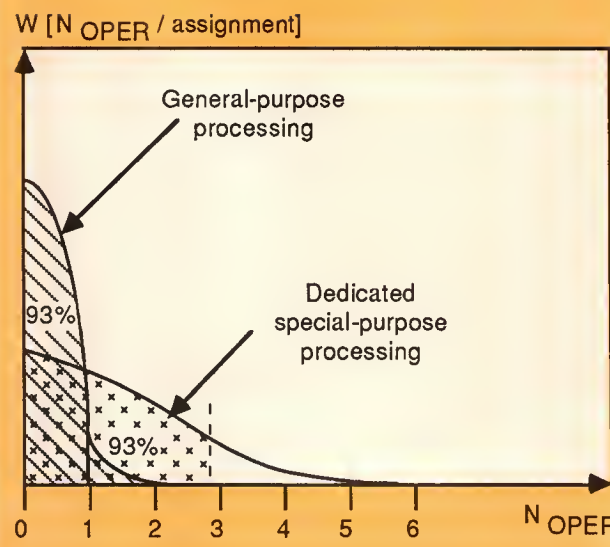


Figure 14. Distribution (W) of the number of operands (N_{OPER}) per HLL statement.

Vertical-migration architecture. Purdue University developed the vertical-migration microprocessor architecture through an ongoing research project.[34] This architecture copes with the large ratio of off-chip to on-chip delays in GaAs by incorporating a multiplicity of simple ALUs onto a GaAs chip. Signal propagation time through these multiple ALUs approaches the instruction fetch time from the off-chip environment. Consequently, the depth of the memory pipeline is much smaller, and it is much easier to resolve pipeline hazards at compile time.[35]

Obviously, this type of architecture is well suited only for applications that are characterized with a high density of ALU operations. Only these applications can keep the multiple ALUs "busy" for most of the time. One example of such application is real-time signal and data processing in digital control, robotics, and signal processing. If these applications are programmed in HLLs, the number of ALU operations per HLL statement is larger than in typical general-purpose programs.

A brief overview of the architecture follows. For further information the interested reader can refer again to the last two references.[34,35] Initial implementation of this architecture occurred in the bit-slice technology. Currently, an E/D MESFET GaAs VLSI design is in progress, and the GaAs WSI implementation is also being considered.

Application environment and the HLL interface. The first phase of this project was to collect dynamic statistics for a large set of HLL programs in digital control, robotics, and signal processing. Figure 14 compares results of this analysis with the results that apply to general-purpose processing.[35]

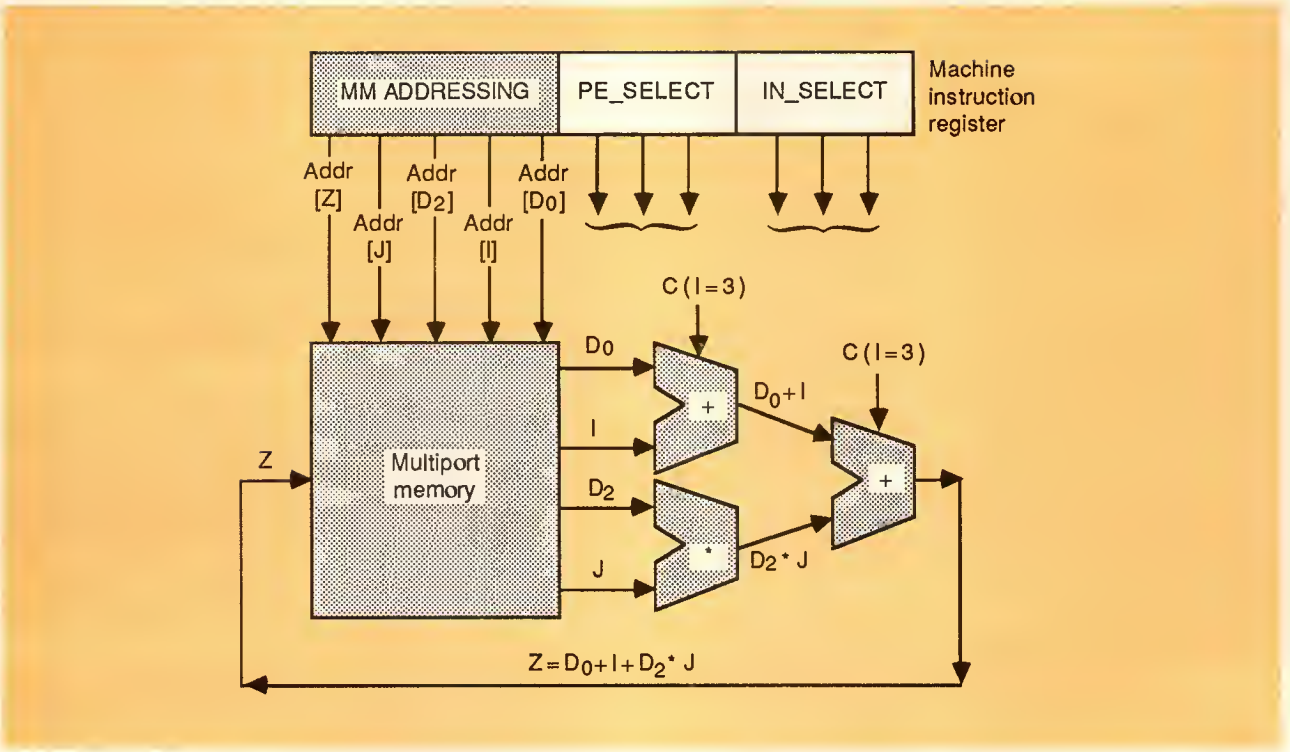


Figure 15. Configuration corresponding to the execution of an arithmetic assignment.

From Figure 14, one can see that 93 percent of assignments in general-purpose processing contain either one or no ALU operation, while in the previously described special-purpose processing, the same 93-percent cutoff occurs at about three ALU operations per HLL statement.[35] Consequently, if three ALUs are incorporated onto the chip, they can be "kept busy" most of the time.

The second phase of the project concerned the development of an HLL language for digital control, robotics, and signal processing. The chosen language is similar to Intel's HLL language for dedicated microprocessor applications, PL/M. It includes only the computing structures like assignments, counter-controlled and condition-controlled loops, call/return, and low-level I/O.

Host machine. Purdue designers chose a Wilkes-type architecture for the host machine, with instructions of 64 bits. The basic characteristic of this host is that it enables "primitive" and the most frequent "nonprimitive" HLL statements to be mapped into a single horizontal machine instruction. Other HLL statements are mapped into multiple machine instructions.

The primitive HLL statement contains only one parameter (for I/O statements and Call/Return statements) or requires only one ALU operation (for Assignment statements and Loop-controlled/condition-controlled control statements). The architecture supports the most frequent nonprimitive HLL statements by incorporating more than one ALU and a multiport register file. Following the findings from Figure 14, designers decided to include three simple ALUs and a register file. That register file is physically a two-port file but behaves as a four-port file since the operands are read in a time-sharing fashion.

Mapping HLL statements into machine instructions. The structure corresponding to the assignment of the form $Z = D_0 + I + D_2 * J$ appears in Figure 15. The machine instruction field MM ADDRESSING defines addresses of the variables stored in the on-chip multiport memory. The fields PE__SELECT and IN__SELECT define the functions of three different ALUs and the interconnection structure among these ALUs. Symbol C(I = 3) indicates that the corresponding ALU can perform eight different functions and is controlled by three lines originating from the field PE__SELECT in the machine instruction register (MR).

Milutinovic et al. presents computing structures corresponding to other relevant HLL constructs.[35]

Comments. This architecture is referred to as the vertical-migration architecture, since it enables a vertical migration of computing from the HLL level

to the machine level with a minimal execution-time overhead. Code generation for this architecture is fairly simple. However, code optimization for increased performance may be relatively complex. In addition to the load-delay and branch-delay pipeline hazards, this architecture includes an ALU-delay hazard. Current research concentrates on the VLSI/WSI implementation issues and the code optimization issues.

We have discussed a number of different systems from bit-slice and related SSI/MSI/LSI components to VLSI microprocessors and a massively parallel systolic array.

All of the processors have one thing in common: Their designs are of the RISC type. As we emphasized several times, the low gate-count restriction of GaAs chips forces designers to either migrate many hardware functions to software (to reduce the transistor count) or to use a multichip approach. In general, however, the multichip approach suffers from high interchip communication costs. If a processor is functionally partitioned into separate parts, such that off-chip communication between parts is confined to a single direction, designers minimize the high cost of interchip communication. Such partitioning, though unacceptable for CPU design, is very feasible for floating-point coprocessors. Due to the inherently sequential nature of floating-point algorithms, designers naturally choose to use a multichip pipelined architecture.[36]

We expect microprocessors designed for GaAs to be clocked at rates of up to 200 MHz, with future rates reaching 1 GHz or more. At such high speeds several problems arise. The lack of high-speed test equipment seriously affects GaAs systems because the microprocessor operations cannot be fully verified. To solve this problem, researchers at the University of California at Santa Barbara are developing a 1-GHz tester for GaAs digital ICs.[37] The lack of memory chips with a sufficiently small access time also poses a major problem. To accommodate the relatively slow memory, either more memory access pipestages must be added to a pipelined processor or the memory access pipestages must be made longer.

Currently, digital GaAs is only used in high-speed supercomputing applications (such as the Cray-3 supercomputer from Cray Research Corporation), in super-minicomputer designs (for example, those by Gould Corporation), in high-speed signal processing applications,[14] or cases in which radiation hardness and tolerance to high temperature variations restrict design factors. In the future, new GaAs fabrication techniques increasing the transistor count per chip while increasing the yield from GaAs wafers would permit many of the systems we described to become more available to commercial markets.

Acknowledgments

The research on which this article was based was sponsored by the NCR Corporation, World's Headquarters, Dayton, Ohio.

References

1. L.E. Larson, J.F. Jensen, and P.T. Greiling, "GaAs High-Speed Digital IC Technology: An Overview," *Computer*, Oct. 1986, pp. 21-28.
2. J. Mick and J. Brick, *Bit-Slice Microprocessor Design*, McGraw-Hill, New York, 1980, pp. 2-189.
3. D.E. White, *Bit-Slice Design: Controllers and ALUs*, Garland Publishing, Inc., New York, 1981, pp. 1-18.
4. "Bit Slice ICs Kick Off Era of Commercial GaAs LSI," *Electronics*, Sept. 18, 1986, pp. 61-64.
5. "VE29G01 Gallium Arsenide 4-Bit Microprocessor Slice," Data Sheet, Vitesse Electronics Corp., Camarillo, Calif., Sept. 15, 1986.
6. Vitesse Electronics Corp., "VE29G10 Gallium Arsenide Microcontroller," Data Sheet, Sept. 15, 1986.
7. W.A. Geideman et al., "GaAs Emulation of Am2901 4-Bit Slice Processor," McDonnell Douglas Corporation Internal Report, McDonnell Douglas Astronautics Company, Huntington Beach, Calif., 1986.
8. M. Hirayama et al., "A GaAs 16-kbit Static RAM Using Dislocation-Free Crystal," *IEEE Trans. Electron Devices*, Jan. 1986, pp. 104-110.
9. J.A. Cooper, M.R. Melloch, and Q.D. Qian, "Development of a One-Transistor Dynamic RAM for the AlGaAs/GaAs HIGFET Technology," Purdue Univ. Internal Report, West Lafayette, Ind., 1986.
10. T.E. Dungan, J.A. Cooper, and M.R. Melloch, "A Thermal-Generation-Limited Buried-Well Structure for Room Temperature GaAs Dynamic RAMs," Purdue Univ. Internal Report, 1987.
11. Y. Nakayama et al., "A GaAs 16 × 16 bit Parallel Multiplier," *IEEE J. Solid-State Circuits*, Oct. 1983, pp. 599-603.
12. T. Furutsuka et al., "A GaAs 12 × 12 Bit Expandable Parallel Multiplier LSI Using Sidewall-Assisted Closely-Spaced Electrode Technology," *Proc. Int'l Electron Devices Meeting*, Dec. 1984, pp. 344-347.
13. T. Vu et al., "A Gallium Arsenide SDFL Gate Array with On-Chip RAM," *IEEE J. Solid State Circuits*, Feb. 1984, pp. 10-22.
14. W. Davenport, "Macro Evaluation of a 3000 Gate Array," *Proc. 1986 GaAs IC Symp.*, Oct. 1986.
15. B.K. Gilbert et al., "The Need for a Wholistic Design Approach," *Computer*, Oct. 1986, pp. 29-43.
16. S. Przybylski, "The Implementation of MIPS," Stanford Univ. Internal Report, Stanford, Calif., Aug. 1984.

17. W.A. Helbig, R.H. Schellack, and R.M. Zieger, "The Design and Construction of a GaAs Technology Demonstration Microprocessor," *Proc. Midcon/85,* Chicago, Sept. 1985, pp. 23/1.1-23/1.6.

18. E.R. Fox et al., "Reduced Instruction Set Architecture for a GaAs Microprocessor System," *Computer*, Oct. 1986, pp. 71-81.

19. *ANSI/IEEE Std. 754-1985, Standard for Binary Floating-Point Arithmetic,* New York, Aug. 12, 1985.

20. T.L. Rasset et al., "A 32-Bit RISC Implemented in Enhancement-Mode JFET GaAs," *Computer*, Oct. 1986, pp. 60-68.

21. V. Milutinovic et al., "Issues of Importance in Designing GaAs Microcomputer Systems," *Computer*, Oct. 1986, pp. 45-57.

22. H. Dietz and C.H. Chi, "A Compiler-Writer's View of GaAs Computer System Design," *Proc. HICSS-88*, Kona, Hawaii, Jan. 1988.

23. C.H. Chi and H. Dietz, "Register Allocation for GaAs Computer Systems," *Proc. HICSS-88,* Jan. 1988.

24. B. Perunicic, S. Lakhani, and V. Milutinovic, "Modelling and Analysis of Stochastic Propagation Delays in GaAs Adders," *Proc. HICSS-88,* Jan. 1988.

25. C. Bauer and D. Meyer, "Processing Element Architectures for Reconfigurable Parallel Systems," PhD work in progress, Purdue Univ.

26. K.J. McNeley and V. Milutinovic, "Emulating a Complex Instruction Set Computer with a Reduced Instruction Set Computer," *IEEE Micro*, Feb. 1987, pp. 60-72.

27. M.A. Overstreet, "Future Advanced Control Technology Study: VHSIC/GaAs," *Interim Report to Industry and Government,* Allison Gas Turbines Division of General Motors, Indianapolis, Ind., Apr. 1987.

28. T.F. Lehr and R.G. Wedig, "Toward a GaAs Realization of a Production-System Machine," *Computer*, Apr. 1987, pp. 36-48.

29. M.L. Morgan, "The Role of Gallium Arsenide in Highly Parallel Symbolic Microprocessors," *Proc. 20th Ann. Hawaii Conf. Ion System Sciences,* Kona, Hawaii, Jan. 1987, pp. 65-74.

30. T. Gross, "Code Optimization of Pipeline Constraints," Stanford Univ., Tech. Report 83-255, Dec., 1983.

31. C.E. Hein et al., "Design of a GaAs Systolic Array for Adaptive Null Steering Beamforming Controller," *Computer*, July 1987, pp. 92-93.

32. J.A. Fortes et al., "A High-Level Systolic Architecture for GaAs," *Proc. 19th Ann. Hawaii Int'l Conf. System Sciences,* Honolulu, Hawaii, Jan. 1986, pp. 253-258.

33. J.F. McDonald et al., "Wafer Scale Interconnections for GaAs Packaging—Applications to RISC Architecture," *Computer*, Apr. 1987, pp. 21-35.

34. V. Milutinovic, "A Simulation Study of Vertical-Migration Microprocessor Architecture," *IEEE Trans. Software Eng.,* Jan. 1988.

35. V. Milutinovic, N. Lopez, and K. Hwang, "A GaAs-Based Microprocessor Architecture for Real-Time Applications," *IEEE Trans. Computers*, June 1987, pp. 714-727.

36. M.S. Vlajnic and T. Maksimovic, "Coprocessor Design for GaAs Technology, *Proc. Euromicro 1987*, Sept. 1987.

37. D.J. Fouts et al., "System Architecture of a GaAs 1GHz Digital IC Tester," *Computer*, May 1987, pp. 58-70.



Harry Vlahos is a graduate student at Purdue University. He is interested in high-performance computer architectures for artificial intelligence and neurocomputing, learning algorithms for neural networks, VLSI computer architectures for GaAs, and parallel processing. He received the BS degree in computer and electrical engineering in 1986 and the MSEE in 1987, both from Purdue University.



Veljko Milutinovic is on the faculty of the School of Electrical Engineering at Purdue University. His current interests include VLSI computer architectures for GaAs, high-level languages, and signal processing. The guest editor for *Computer* magazine's special issue on GaAs microprocessor design, he also serves on the Euromicro board of directors. Milutinovic received a PhD degree in 1982 from the University of Belgrade, Yugoslavia.

Questions regarding this article can be directed to Veljko Milutinovic, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907; (317) 494-3530.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 153 **Medium** 154 **High** 155

The Balance Multiprocessor System

Shreekant Thakkar, Paul Gifford, and Gary Fielland
Sequent Computer Systems

By pooling 30 VLSI processors and sharing memory, this powerful computer can speed up a standard floating-point program by a factor of 27.4.

Balance is a shared-memory, tightly coupled multiprocessor system, whose architecture, operating system, and performance we describe here.[1] Balance can contain two to thirty 32-bit microprocessors with an aggregate performance of up to 21 million instructions per second (MIPS). Each processor has a private cache as well as a small local memory to hold frequently used kernel routines. The system features a high-bandwidth pipelined bus, up to 28M bytes of main memory, a diagnostic and console processor, up to four IEEE 769 (Multibus) adapters, an IEEE 802.3 (Ethernet) LAN interface, and an ANSI Small Computer Systems Interface (SCSI). Dynix, a multiprocessor operating system supporting both the 4.2 BSD and System V Unix environments, manages Balance. It provides transparent support for multiprocessing as well as tools and libraries for developing parallel applications.

The processor subsystem

The Sequent Balance 8000 and 21000 systems are members of a family of products that implement a scalable processor pool architecture (Figure 1 on the next page). Each system is composed of a pool of two to 30 processors. As can be seen in Figure 2, each processor in this architecture is itself a subsystem, packaged two per circuit board. A subsystem includes three VLSI parts from the National Semiconductor Series 32000 family, a 32-bit 32032 CPU, a 32081 hardware floating-point unit, and a 32082 paged virtual memory management unit.[2]

The 32-bit CPU architecture is well suited to applications that use 32-bit integer and address calculation operations. The floating-point unit is also important for the extended precision, dynamic range, and performance often required in engineering and scientific applications. A paged memory management unit provides hardware support for a fully associative translation lookaside buffer and a two-level page table that provides up to 16 megabytes of virtual address space for each process.

The cache. Each processor subsystem includes a physical address cache memory to provide zero-wait-state performance while minimizing bus traffic.[3] Caching, six times faster than accessing memory, matches the high-speed CPU with the lower speed memory. The Balance two-way, set-associative cache provides 8 kilobytes of high-speed buffer memory to

0272-1732/88/0200-0057$01.00 © 1988 IEEE

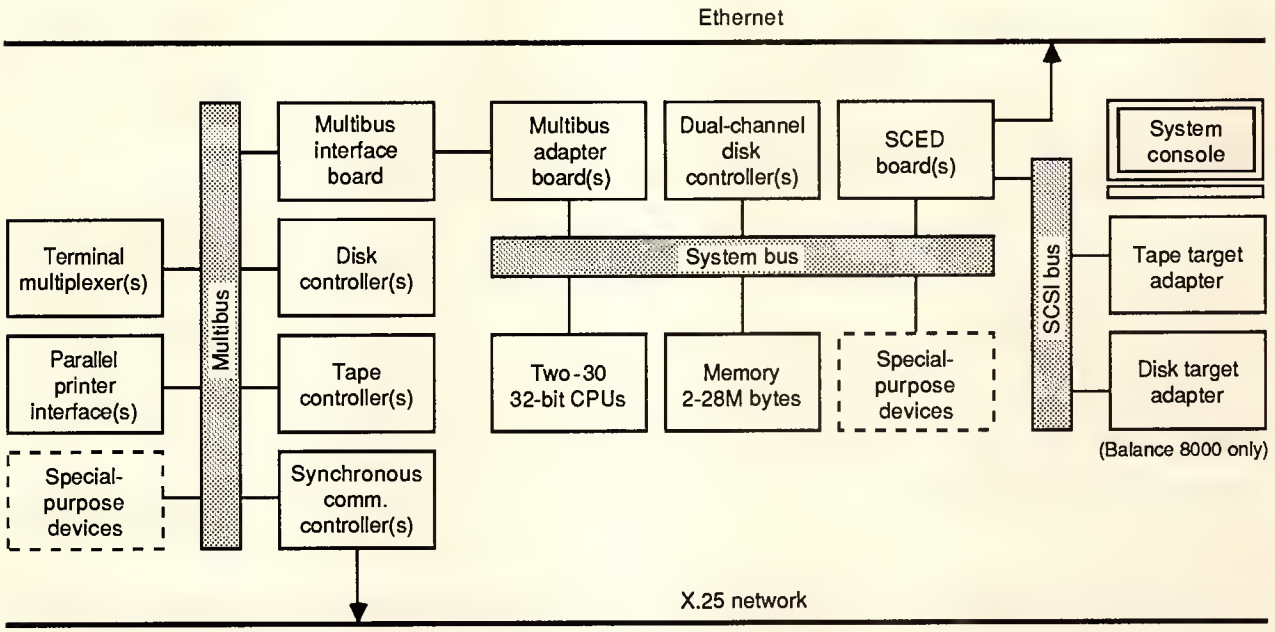


Figure 1. An architecture with shared memory, a common bus, and a processor pool.

store recently accessed instructions and data.[4] The cache, rather than main memory, satisfies subsequent requests for this information, significantly enhancing performance while conserving both bus and memory bandwidth.

Our designers selected an 8-byte line size for the cache because it represented at that time the best trade-off between hit ratio and bus traffic. This line size makes the 8-byte memory transfer a particularly important system operation. Simulations and measurements indicate that the 8-byte line (block) size yields an effective hit ratio of 95 percent for most integer applications.

Designing a cache for a processor pool multiprocessor is difficult. Since each cache datum represents a copy of the actual memory datum, the system must ensure that all such copies, including the original, remain identical. To ensure that the main memory datum is updated whenever a cached copy is updated, the Balance system employs a nonallocating write-through cache consistency protocol. Each write cycle goes through to the bus and memory, in addition to updating the cache. Though writes to memory represent only about 10-15 percent of all processor operations,[4] designers still worried that performance would be impacted by the processor waiting for the completion of each such bus write cycle. To avoid this potential performance impact, they implemented a write buffer. This write buffer provides temporary storage, allowing the processor to quickly deposit the write address or data and proceed while the write buffer independently completes the main memory cycle. For successive writes, the second write holds up if the first write is not completed.



The Sequent Balance system.

The write-through protocol satisfies the consistency requirement of keeping main memory data consistent with cache data. However, additional intelligence must be provided to keep data coherent among the caches. Consider the case in which each of processors

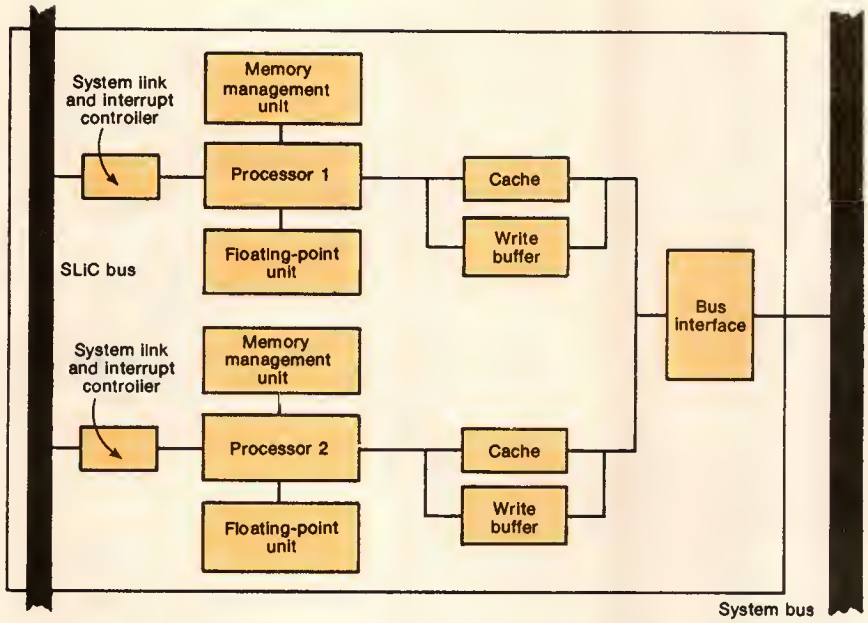


Figure 2. Processor subsystems are packed two per circuit card.

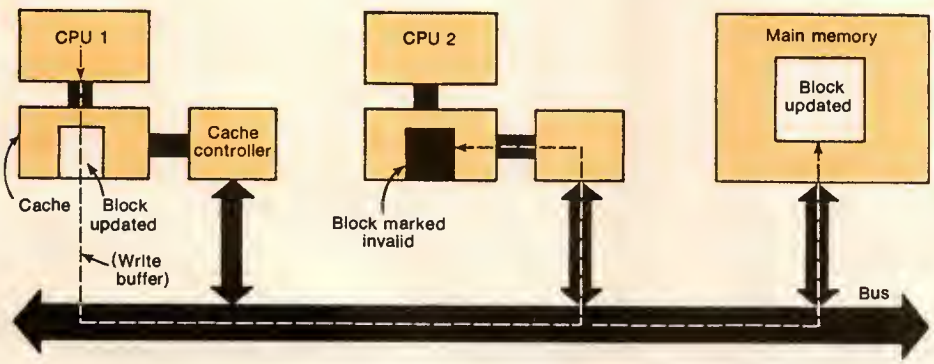


Figure 3. Cache coherency maintained using write-through operations and bus-watching logic.

P_1 and P_2 have recently read a given memory datum X, thereby depositing X into each respective cache. If one of the processors, say P_1, were to write a new value to datum X, we would have to ensure that P_2 does not subsequently use the old (stale) X datum residing in P_2 cache.

With the Balance system a set of bus-watching logic at each cache continuously monitors write cycles on the system bus, comparing the write addresses to the internal cache state so it can detect hits to cache entries (Figure 3). This logic consists of a duplicate set of cache address tag directories. When a hit occurs in the bus-watching logic, the cache controller invalidates the affected entry. Thus, in the example above, the cache of P_2 would have invalidated its X datum entry when P_1's write cycle appeared on the bus. If P_2 subsequently read datum X, it would get a cache miss and fetch the valid main memory copy of X.

System Link and Interrupt Controller. Sequent developed the System Link and Interrupt Controller chip to address a number of the multiprocessor design issues.[5] One SLIC chip couples with each major component in the system (for example, processors, memory controllers, I/O controllers), providing support for interrupt distribution, low-level mutual exclusion, and configuration and error control. A major goal of the SLIC design was to either remove many of these concerns from the software or provide sufficient hooks to simplify the problems wherever possible. Sequent also desired to simplify the system bus, lowering its cost while maintaining high performance and reliability.

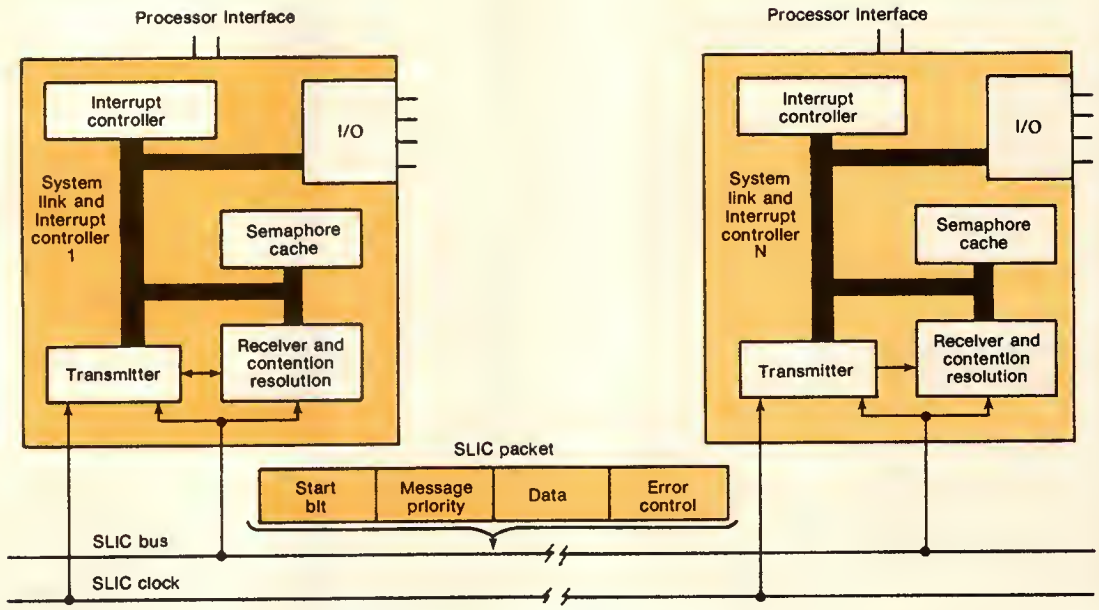


Figure 4. Low-level communication and control in the System Link and Interrupt Controller.

As Figure 4 shows, SLIC chips in the system communicate with a simple message-passing protocol, using two lines from the system bus for a bit-serial, wired-OR media (the SLIC "bus"). The SLIC bus derives its clock from the system bus but otherwise operates asynchronously. Thus SLIC message traffic becomes transparent to the system bus and does not impact bus bandwidth. SLIC bus access is arbitrated using wired-OR resolution. In case of contention the higher priority message automatically wins the arbitration (priority is part of the message encoding). Each SLIC has a system-unique "number," derived from its host-board position in the system backplane; this number can break a tie during arbitration. The arbitration technique is unusual in that contention for the bus does not degrade throughput.

An SLIC provides two basic kinds of interrupts, maskable and nonmaskable, which correspond to the interrupt request lines present on most microprocessors. The type of interrupt and the priority level (bin number) are encoded in the command field of the SLIC message. Interrupts are further characterized by the way they are directed: to a particular SLIC (*directed* interrupt) or one of a group of SLICs (*group* interrupt. The selection of group is an initialization parameter to the SLIC.) All SLICs in a group attempt to accept the interrupt, subject to their current state; at most one of the SLICs accepts the interrupt, using the responding SLIC address as final arbitration resolution. To allow some software control over arbitration in accepting group interrupts, the SLIC supports an 8-bit register whose upper 5 bits are fully software controlled (the SLIC randomly generates the lower 3 bits).

When an SLIC accepts an interrupt, it copies the message data into an internal register, and asserts the appropriate processor interrupt line. The SLIC then refuses to accept another interrupt until the operating software acknowledges it has started its interrupt handler.

Most operating systems accumulate per-process execution statistics, as exemplified by the Unix notion of user and system time reported by the time command. On a multiprocessor system a single system clock could support this only by broadcasting a periodic interrupt and ensuring that all processors accept it. It is more convenient to have each processor generate its own clock. This scheme also has advantages with respect to cache usage: per-processor clocks can be mutually asynchronous, a state that avoids forcing all processors to change their cache context at the same time, thus spreading the system bus load more evenly. To support this scheme, an SLIC implements a programmable internal timer.

Each SLIC implements a set of 64 binary semaphores, called gates, and supports a set of SLIC commands to atomically test and set them. Each SLIC contains a copy of the value of each of the 64 gates at all times. If an attempt is made to "lock" a gate, the SLIC uses its local copy of the gate value to determine if an SLIC-bus message is necessary. If the gate is already locked, the SLIC sets a status bit and sends no message. If the gate is unlocked, the SLIC

sends a message to attempt to lock the gate. All SLICs see this message and set their local copy of the gate to locked status. The arbitrated SLIC lock-gate message ensures that at most only one SLIC sends a lock-gate message at any point in time, thus assuring the atomicity of the operation.

Another SLIC message unlocks a particular gate, and all SLICs clear their internal copy of the gate. The gate number is encoded in the request-message data field of the message.

Since gates are used for low-level mutual exclusion in the operating system, any "write-behind" data must really be in system memory before actual release of a gate. For example, data modified while the gate was locked might be in the processor write-buffer on its way to system memory but not yet there. To avoid race conditions that could result from this situation, an SLIC doesn't actually send the unlock-gate message until the processor write buffer empties (the write buffer exports a status pin for this purpose). Thus when software releases a gate, it has automatically guaranteed the consistency of system memory.

An SLIC supports a 256-byte local address space, called *slave registers*. The SLIC itself, or any other SLIC in the system using two messages directed to a particular SLIC, reads or writes these variables. One specifies the address and the next reads or writes the data. Each hardware board in the system implements a subset of these addresses for its SLIC.

Access to an SLIC occurs via 17 byte-wide locations in the local processor address space. Some of these locations are simple read and/or write registers, for example, local SLIC interrupt mask, arbitration priority register, local SLIC number (provides processor identification as well), timer-control registers. Sending a message involves loading several registers with the message data, encoded destination address, and any other necessary data, then loading the command register to start the command. The software then waits, looking at a status bit to see when the command completes. Once the task is completed, the status register contains bits describing aspects of the result: Did the message get sent, was it accepted, were there any errors, etc. Software can retry the command and return a success/fail indication, or whatever makes sense. The programming model relies on a spin-waiting technique for results since each SLIC command completes quickly with its status; servicing an interrupt would impose unreasonable overhead.

The system bus. The system bus is a critical element of the system. It provides software-transparent, symmetric access from all processors to all system resources, including main memory and I/O subsystems of widely varying access latency. The bus was designed with careful regard to the memory bandwidth required by a pool of high-performance processors and I/O devices.

A global, 10-MHz synchronous bus interconnects all processors to all other resources. This bus provides the necessary performance with a 32-bit, parallel, time-multiplexed address and data path, as well as a set of control paths. The bus interface has been implemented using discrete components. On newer designs, the bus controller function can be implemented in a single 1.5-micrometer, CMOS gate array.

A split request/response protocol optimizes the desirable bandwidth over the bus data path. In this protocol, memory operations are pipelined, and individual requests and responses can be interleaved onto consecutive bus cycles. Separate request pipelines (queues) for memory read, memory write, and I/O subsystem requests avoid the relatively long latencies associated with processor accesses to storage on slower Multibus I/O adapters.

The protocol involves splitting responses from requests, so that the bus is occupied only for those cycles necessary to transmit the request and response information. Storage access itself does not take bus time. Instead, it occurs in parallel with bus traffic from additional requests and responses. The write response is signaled "out of band," which is to say that it has its own dedicated set of control wires on the bus. This feature frees the main 32-bit data path for other traffic.

Figure 5 on the next page presents an example that may clarify the parallelism inherent in this protocol. Processor P_1 emits a 1-byte read request destined for the Multibus. It transmits in one 100-nanosecond bus cycle and queues the request on the Multibus coupler. Processors P_2 and P_3 both transmit 8-byte read requests to primary storage in the following two bus cycles, freeing the bus for further traffic. The memory controller queues the requests and starts the access. When the 8 bytes of data requested by processor P_2 are available, the memory controller takes two bus cycles to return them to P_2, followed immediately by the two bus cycles used to return the 8 bytes of data to P_3. The bus is then free again until some time later when, depending on the speed of the addressed Multibus device, the Multibus coupler obtains its data from the device and occupies the bus to respond to processor P_1. Thus the bus traffic is decoupled from memory access times. The combination of this bus together with the Sequent memory controllers, which can be interleaved for performance, result in an effective sustained data bandwidth of 26.7M bytes/s.

In addition to the bandwidth considerations, a multiprocessor bus implementation also requires considerable attention to arbitration, congestion management, and error control and recovery. The system bus employs a centralized multilevel arbiter. The arbiter

provides multiple priority levels to ensure low latency for mechanical mass-storage devices whose performance would be seriously impaired if data transfers were not accepted in real time. All processors however share a given priority level, and within that level the arbitration circuitry guarantees fairness through a round-robin algorithm. This fairness very importantly ensures that, even under a very heavy load, no condition exists in which a given processor would be totally starved for bus access.

A related issue in split-response bus protocol systems is that of congestion management and flow control. Instantaneous heavy loads on the bus can result in full request pipelines or queues on responders. In some systems, the flow control mechanism that keeps these queues from overflowing involves a negative, or NAK, acknowledgment response. However, such a system can degrade nonlinearly as a function of an offered load since the bus is further degraded by nonproductive request-NAK cycles.

The Balance system manages congestion with a distributed flow control mechanism. In this scheme, each requester maintains status information about each relevant queue. Requests are not propagated onto the system bus unless the request is guaranteed a slot in the addressed queue. Thus even under a very heavy load, each and every bus cycle productively responds to the offered load.

Error control and recovery is always a difficult issue, but it's even more difficult in a multiprocessor due to the multiplicity of bus agents and the high degree of system concurrency. The bus provides parity as well as other error detection facilities to maximize the probability of detecting errors in and around the bus system. Whenever a serious error occurs, the hardware records the identities of the parties involved and freezes the bus so that the error does not propagate and create further damage. Upon the detection of such a condition, the diagnostic processor takes control of the system and uses the SLIC mechanism to investigate the problem. The recovery software typically resets the system, runs selected confidence tests, perhaps disables the faulty module, and reboots the system on the remaining validated hardware.

Balance systems are not fault-tolerant systems. Hence any processes that are running whenever the system error occurs (called panic) are lost, unless application-specific steps have been taken to checkpoint the processes, allowing system restart.

The memory subsystem. The memory subsystem stores all resident code and data for the entire system and must meet all requests for code or data that are not satisfied by processor local caches. Thus it is critical that the memory system provides a high-access bandwidth to minimize the performance impact of memory operations on the system.

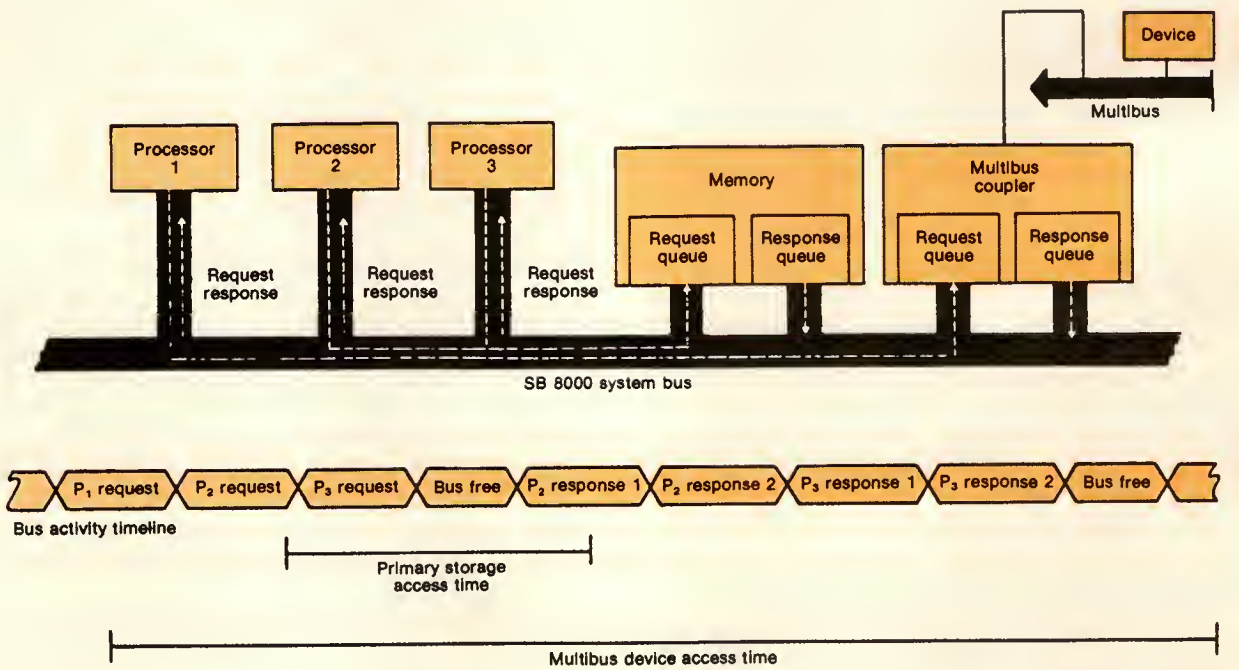


Figure 5. Pipelined packet protocol optimizes bus bandwidth.

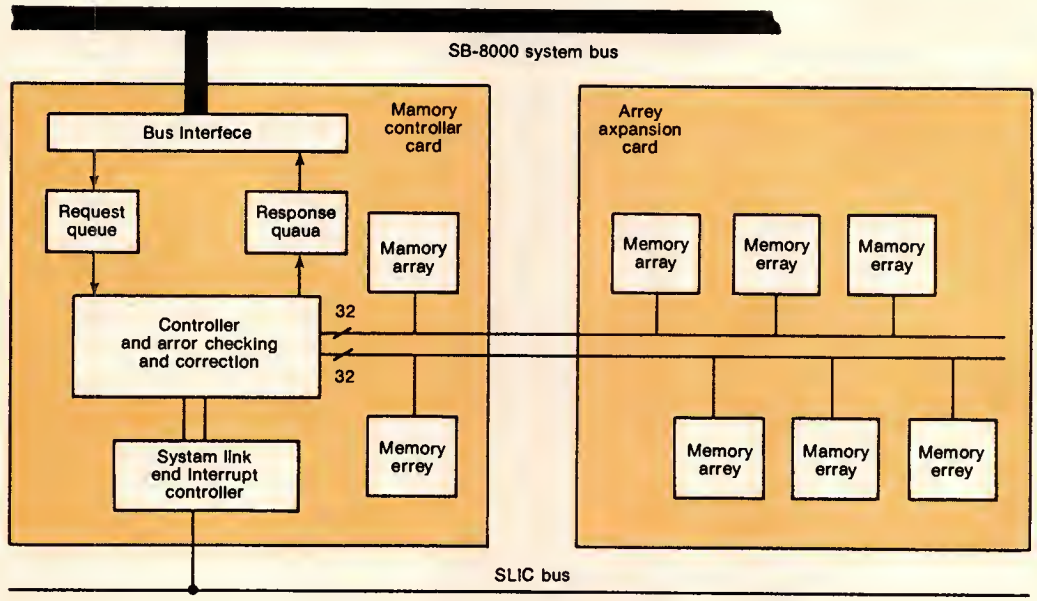


Figure 6. Memory bandwidth optimized via interleaving and pipelining.

To meet this requirement, most memory read and some memory write operations transfer in 8-byte lengths. Combining this larger transfer size with the split transaction bus protocol results in a pipelined memory subsystem composed of a number of elements (Figure 6). Requests arrive from the bus interface and are stored in the request queue. Simultaneously the controller can be cycling the 64-bit-wide arrays to access data while detecting and correcting errors. Also in parallel, the response queue can be emitting responses from previous requests. This pipelined operation allows a single memory controller to support a sustained data transfer rate of 8 bytes every 300 ns. The SLIC circuit on the memory controller card reports error information, identifies the memory module and its configuration to the autoconfiguration software, and allows the configuration software to set attributes (base address and interleave factor) on the card.

The I/O subsystem. Though many applications are often characterized by their compute-intensive requirements, multiprocessor systems must also provide a balanced I/O capability. To satisfy this requirement, Balance supports one or more SCSI/Ethernet/Diagnostic (SECD) subsystems and dual-channel disk controller (DCC) modules.

Each SECD card includes an instruction-set-compatible 32016 processor, which interprets channel commands and drives the on-board I/O adapters (Figure 7 on the next page). The processor on the first channel board also serves as the diagnostic and console processor, taking advantage of the fully self-contained environment provided on this card. Thus the channel/diagnostic processor operates normally, even in the event of hard errors in other portions of the system like the bus, memory, or processor pool.

The supported I/O adapters include an IEEE 802.3-compatible Ethernet interface and an interface to the ANSI Small Computer System Interface mass-storage bus. This use of industry-standard interfaces makes it easy for a customer to mix and match mass-storage devices, selecting from a growing set of compatible devices. To provide high performance, Balance supports these adapters with individual surge buffers (first-in, first-out buffers) and direct memory access (DMA) controllers designed to support the 8-byte bus transfer facility. The board supports the sustained data transfer rate of both adapters simultaneously.

The DCC high-performance disk controller drives up to eight disks on two independent channels. Each channel can perform independent, overlapped seeks on all four of its drives simultaneously. Each channel transfers data at 24M bits/s. The two channels can also transfer data simultaneously, each at full transfer rate. Double command buffers in the system software allow two disk requests per drive to be active at any time, achieving efficient performance in back-to-back reads and writes. Total throughput of a DCC with multiple drives under the Unix file system is over 3M bytes/s.

The IEEE 796 (Multibus) bus coupler provides a separate interface to add custom hardware or one of the many commercially available Multibus-compatible, special-purpose I/O adapters. Direct

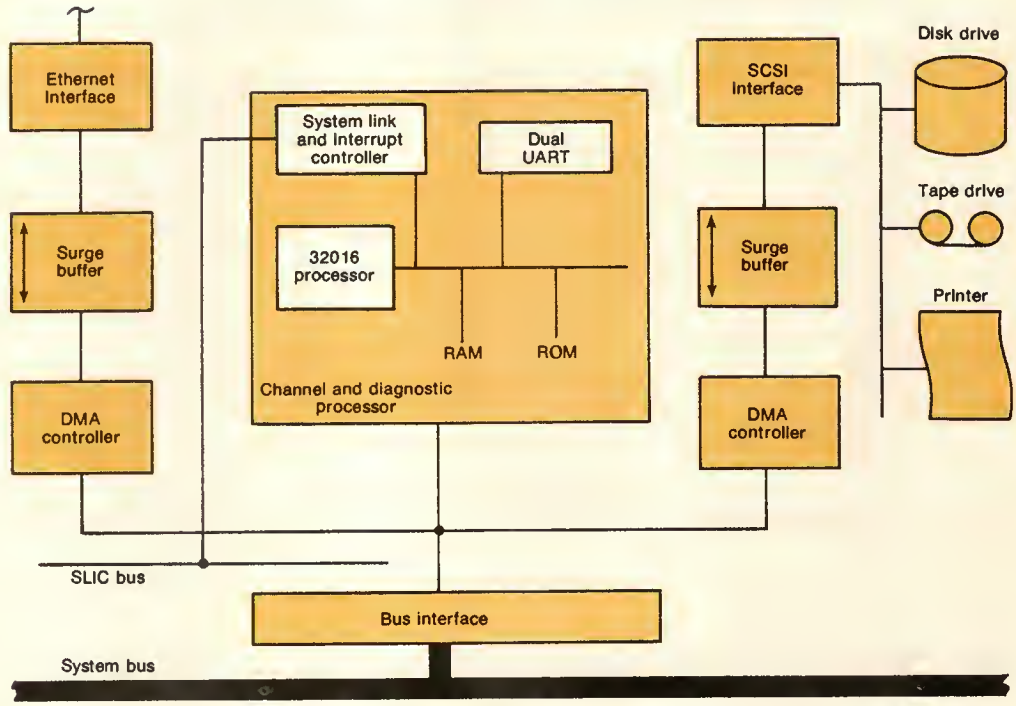


Figure 7.
SECD controller.

transfers from the system bus to Multibus, as well as transfers from the Multibus to the system bus, are supported. The bus coupler provides address mapping and protocol conversion services. The system supports up to four such Multibus couplers for systems with a large I/O complement. Couplers for other low-complexity buses like the VMEbus could be built in a similar manner.

Dynix operating system

Even though the Unix philosophy and model are based on multiprogramming, the standard implementation supports a single-processor architecture. On such a system, concurrency appears *apparent*, not *real*. Since only a single processor performs, the processes are time multiplexed (time sliced) through the processor. In a multiprocessor several processes often simultaneously execute the operating system code within the kernel on several processors. Thus many simplistic single-processor assumptions do not hold, and multiple, simultaneously executing processors need to fully share the kernel.

Dynix, Sequent's multiprocessor implementation of Unix, is the Balance operating system. Dynix required enhancements in five major areas of the operating system model: mutual exclusion, interrupt distribution, process scheduling, shared-memory management, and virtual memory management. Use of the SLIC made some of these enhancements possible.

The SLIC subsystem resolves problems regarding processor synchronization, dynamic interrupt distribution among processors, uniprocessor driver support, interprocessor communication, dynamic load balancing of processes among processors, and dynamic system configuration.

Processor synchronization. A multiprocessor system requires a fast, efficient synchronization primitive. The SLIC provides this function in the form of a cache of 64 single-bit gates. Gates, logically equivalent to the test-and-set primitive, are spin oriented. That is, the process loops request the gate until they acquire it. In most machines an interlock signal on the system bus and/or in the memory controller itself implements the test-and-set function. This process requires extra complexity in the system bus architecture.

The use of SLIC gates as the synchronization primitive offers two main advantages. First, gate operations proceed via the SLIC bus. Second, since mutual exclusion is achieved via the gate mechanism, the system bus and memory architecture can be simpler. Since the SLIC bus is separate and asynchronous to the system bus, SLIC gate accesses do not use the system's bus or memory bandwidth. Since each SLIC knows the status of all gates, spinning and watching the local SLIC status register occur simultaneously and not across the SLIC bus. Thus, gates do not adversely affect the SLIC bus bandwidth. This technique relieves one of the performance

bottlenecks found in previous multiprocessor systems carrying a heavy load. The mutual-exclusion solution allows the system bus to be built at a lower cost, with performance maximized and a higher degree of reliability.

The Dynix system uses three types of mutual-exclusion primitives built from gates: spin locks, counting semaphores, and the direct use of the gates themselves.

The SLIC chip directly implements the gate, the lowest level primitive. Gate acquisition is the fastest among the three types of mutual-exclusion primitives. But since a finite number of gates exist, they are only used in the most time-critical regions. For example, a gate synchronizes the processors' accesses to the run queue. Gates, because of their spin-oriented nature, are held for only a short period of time to minimize contention. However, when contention occurs, the processor waiting for a busy gate consumes no system or SLIC bus cycles.

Balance designers defined locks as multiplex gates only. Whereas only 64 gates exist, the number of locks is unlimited. A lock uses a shared-memory location to encode the lock state (locked or unlocked) guarded by a gate. The gate is acquired only to manipulate the lock variable (that is, to set the lock). Therefore, the lock operations are also atomic. A single gate can synchronize accesses to many locks. For example, changes to the state and flag variables for a process in the process table must be performed atomically. First, the process acquires a per-process lock. If the system is configured for 500 processes, 500 such locks exist. However, only 10 gates might be assigned to synchronize the manipulation of those locks. Locks, like gates, are spin oriented and are used to guard short, critical regions. Since each processor is equipped with a bus-watching cache, the system stores the instructions and data accessed while waiting for a lock in the cache. Therefore, the wait for a busy lock does not consume system bus cycles.

In a uniprocessor Unix implementation, the set priority level (SPL) mechanism provides the synchronization necessary between interrupt level and base-level access to shared data structures. Specifically, the base-level code must raise the interrupt priority level of the processor to block out interrupt routines before executing a critical region. In a multiprocessor architecture a form of interprocessor synchronization (gates and locks) is necessary in addition to the SPL mechanism. Base-level code must raise the processor interrupt priority level when acquiring a lock to avoid the deadlock that occurs when an interrupt routine on the same processor attempts to acquire the same lock.

The highest level mutual-exclusion primitive used is the counting semaphore. This semaphore consists of a counter and a wait queue. A gate guarantees the atomicity of the semaphore manipulations. Semaphores block access while waiting for an event or when the critical region guarded by the semaphore is very long.

Semaphores completely replace the conventional Unix sleep/wakeup mechanism. In the sleep/wakeup model all waiters are awakened, and they all compete for the same resource. These processes proceed to race to acquire the resource. One process wins and acquires the resource; the others go back to sleep again. On a uniprocessor, scheduling the first process so it receives the resource eliminates this problem. By the time the rest of the processes are scheduled, the resource is probably available to them. However, in a multiprocessor the awakened processes could all be scheduled simultaneously, causing unnecessary context switching in which the context switches tend to invalidate the processors' caches. This scheme causes heavy system bus activity while the caches are being refreshed. In general, semaphores are more appropriate in a multiprocessor because they provide more structure and eliminate this type of unnecessary context switching.

Interrupt distribution. Perhaps the most important function of the SLIC is that of interrupt control and distribution. To eliminate the potential overloading of a single processor with the entire system's interrupt load, the SLIC provides dynamic interrupt distribution.

The SLIC subsystem handles all device interrupts in the system. The subsystem maps interrupts from Multibus and SCSI bus peripherals to SLIC interrupt messages. All such SLIC interrupts are mapped to SLIC bins 1 to 6. Each bin has a potential for 256 interrupt vectors (unique SLIC messages). Although this scheme provides good flexibility, it is undesirable to build an interrupt vector table for all possible vectors. Programmed SLIC interrupts tell each module (processor, controller) on the system bus at system initialization time which SLIC bin and message data to send for each interrupt. Interrupt vector tables can then be optimally sized at system initialization time.

Every SLIC on the SLIC bus responds to interrupts directed at the individual SLIC *id*. In addition, each SLIC corresponding to a processor is programmed to respond to a destination group id. All device interrupts are directed to this group id. When an interrupt is injected into the system, the SLICs in the processor group arbitrate among themselves to determine which accepts the interrupt. Once accepted, the subsystem masks the bin on the accepting SLIC until completion of the interrupt handler so that this SLIC no longer arbitrates for other interrupts in that bin. The acceptance of an interrupt by one SLIC in a group does not inhibit another SLIC in the same group from accepting another interrupt from the same or other bin. This technique allows multiple device interrupts to be serviced simultaneously.

Directing interrupts to a group of processors has advantages. First, interrupts will be dynamically distributed among the group of processors. Thus, no single processor nor statically defined group of processors become a bottleneck. Second, processors can be brought on line or taken off line dynamically without the loss of any interrupts. Finally, this architecture allows improved interrupt latency over that of a uniprocessor.

An SLIC arbitrates for interrupts based on its local priority register. The lower the priority, the more likely the SLIC will win the arbitration. The kernel sets the local priority register to reflect the scheduling priority of the process currently running on the processor. Idle processors set the SLIC priority register to the lowest possible value. The idea is to have the processors running the least important processes handle most of the interrupt load. Thus, the interrupt load is automatically and dynamically off-loaded from the processors doing the most important work.

Interprocessor signaling. The SLIC solves another multiprocessor issue with interprocessor signaling. A processor may need to signal another processor when that processor has some task to perform. An example is a preemptive rescheduling nudge. Programmed interrupts through the SLIC implement this signaling. With the SLIC a processor can send another a normal, maskable interrupt (Bin 1-7); a nonmaskable interrupt; or a software interrupt (Bin 0).

Process scheduling. The process scheduling technique in a symmetric multiprocessor is conceptually similar to that in a uniprocessor. Whereas the uniprocessor scheduling policy is to always execute the highest priority runnable process, the multiprocessor scheduling policy is to always execute the set of N highest priority runnable processes, where N is the number of processors. The multiprocessor dispatching model must:

- Provide for dynamic load balancing.
- Dynamically adapt to N processors, for $N >= 1$.
- Avoid unnecessary process migration and context switching. Each dispatch causes heavy bus activity until the processor memory cache is refreshed.
- Allow for the dynamic starting and stopping of processors.

The Dynix scheduler uses a single priority-ordered queue of runnable processes (RunQueue). Processes in the RunQueue make no processor-specific distinction, and normally no static binding of processes to processors occurs. Since all processors are identical, any process (whether in User or Supervisor mode) may run on any processor. The symmetric shared-memory architecture allows for easy implementation of the dynamic load balancing requirement. Since all processors are identical and all process state information, code, and data reside in a common shared memory, process migration is trivial.

The basic dispatching algorithm is fairly simple. Each processor upon entering the dispatcher merely removes the highest priority process from the RunQueue and executes it. A single SLIC gate synchronizes accesses to the RunQueue. If no work exists, the processor executes its idle loop. The idle loop spins, testing the RunQueue for runnable processes. Note that this testing requires no SLIC gate synchronization, since it is sufficient to merely detect change in the RunQueue status. Once change is detected, the idle loop returns to the dispatching loop. The idle loop takes advantage of the bus-watching cache; therefore the spinning consumes no system bus cycles nor SLIC bus cycles.

Besides voluntary rescheduling, processes can be preempted by either time-slicing or runnability of a higher priority process. A time-slicing interrupt enters the system periodically to cause the accepting processor to determine if a running process should be preempted by a higher or equal priority process in the RunQueue. If it should, a directed software interrupt nudges the appropriate processor to reschedule. A process often becomes runnable as a result of a device interrupt. The processor accepting the device interrupt determines whether the awakened process should preempt a currently running process. If it should, it nudges the processor running the lowest priority process to reschedule.

Dynix loads the SLIC's priority register with the process priority level whenever a context switch occurs. The result is that the CPU executing the lowest priority process fields interrupts.

System configuration control. In addition to its role as an interrupt controller, the SLIC subsystem provides a convenient, simple, and reliable communication path among modules (processors, controllers, bus adapters). This communication path determines system configuration, configures and deconfigures modules, brings processors on line or takes them off line, and functions as a communication path for error management information.

SLIC slave registers supply the information necessary to support these diverse functions. Each SLIC has the capability of either reading or writing any other SLIC's slave registers via SLIC messages. In addition to reporting status, SLIC slave registers also serve as command registers for their respective modules. The functions attached to these registers depend on the module. For example, slave registers on the memory controller report error information, identify the memory configuration (64K or 256K chip technology), and set various configuration attributes of the controller (base address, interleave factor, and so on). Also, remote SLIC messages directed at the

desired processor SLIC bring processors on line and take them off line.

To facilitate system configuration, all modules use a common subset of SLIC slave registers. These registers include configuration information such as module type (processor, memory) and revision level. With the SLIC slave-register mechanism to communicate configuration information as well as set configuration attributes, SLICs have no need for switches or wire-wrap stakes on the modules on the system bus.

The system's power-up firmware takes advantage of the SLIC bus to probe for the presence of hardware modules. It then builds a complete autoconfiguration table in a known place that can be used by diagnostics, the Dynix operating system, or any other stand-alone program. When another processor board is plugged into the backplane, the power-up firmware automatically adds this resource into the configuration table. No changes to the software are necessary, and a single Dynix binary controls a wide variety of hardware configurations. This scheme simplifies the task of field upgrades and maintenance.

Shared-memory management. As described earlier, the kernel data structures are shared through gates in the SLIC. Thus for protection reasons users cannot directly access the SLIC. A separate set of test-and-set variables, called atomic lock memory, allows users to share data. This memory, located in the I/O address space, can be mapped to the user's process address space to provide convenient and fast access to these variables. These locks can be used directly or as a basis for constructing higher level software synchronization mechanisms. The Dynix Parallel Programming Library supports some higher level synchronization mechanisms.

Virtual-memory management. Balance designers redesigned the Berkeley Unix system of virtual-memory management. Berkeley Unix uses a *global* model[6] in which a special process called a *pageout daemon* handles certain paging operations for all processes. This model assumes that no other process can execute when pageout daemon is executing. However, this assumption is invalid for a multiprocessing system. The Dynix virtual-memory system uses a *local* model,[7] similar to the one used on the VAX/VMS system in which each process has a greater role in its own paging activity.[8,9]

Applications

The Balance system takes advantage of parallel processing in two ways, implicitly (transparent to the user) and explicitly (through user programming).

Perhaps the easiest win for multiprocessing is to take advantage of the implicit parallelism in an existing multiuser, network server, or multiprogramming environment. In this environment, there generally are a large number of Unix processes ready to run as a result of user activity. In Unix 4.2, this number is reported as the system load average. Thus if the number is 10, 10 processors could be usefully employed to improve system throughput. The mechanism of transparent dynamic load balancing in Dynix ensures that at any point in time, Dynix schedules the highest priority ready-to-run process onto a processor. For instance, commercially available database management systems employ one to two Unix processes per logged-on user, plus a set of daemon processes for things like read-ahead buffering. Users have demonstrated near-linear improvement for throughput of compute-intensive database queries when adding processors to the pool.

Though the implicit techniques are perhaps the easiest win, Unix also frequently makes it possible and easy to explicitly adapt an application to exploit latent parallelism. For instance, large software systems on Unix are composed of hundreds of hierarchically organized, separately compilable files. *Makefiles* in conjunction with the Make utility define the instructions for building the software system. These instructions define which files need to be macroprocessed, compiled, linked, formatted, and so on.

By adding a single background character (&) to a Makerule, the Sequent 12-processor Balance 8000 realized a factor-of-seven reduction in build time. For instance, the current Dynix operating system, including both 4.2 BSD and System V command libraries, can be built in approximately three hours using the parallel Make utility. Standard Make requires over 22 hours to build the equivalent on the same system.[10]

The techniques described above exploit *large-grain* parallelism. That is, the size of the executable units run in parallel is relatively large—for instance, a complete compilation. The *small-grain* parallelism found in applications such as logic, fault, and circuit simulation can be exploited on the Balance system by explicitly expressing parallelism in the program. To avoid Unix tasking and scheduling overhead for such applications, Dynix provides a microtasking facility. This facility allows several microprocesses to be mapped onto a single Dynix process. Such a facility can be also be written by users to support their own microtasking environment. This capability allows users to provide a scheduling policy that is suitable for their application. For instance, the Ada language on the Balance system has its own microtasking facility.

Parallel programs are difficult to debug when using conventional debuggers that monitor only one process at a time. The Sequent Pdbx parallel debugger allows the programmer to execute a parallel program in a controlled environment in which all the streams of execution can be monitored.

Table 1.
CPU comparison using the Dhrystone benchmark (Dhrystones/second).

Description	B8000 One copy	B8000 Cum.—12 copies	VAX 11/750
C	1104	12,793	872
Pascal	914	10,068	745

Performance

Several measures of performance exist for a multiprocessor system such as the Balance system. First, we can compare each CPU with other uniprocessors. Table 1 lists a comparison using the Dhrystone benchmark; the Dhrystone test attempts to simulate a typical, single, CPU integer-application instruction mix.[11] This benchmark, small and entirely contained in cache, really measures the processor performance rather than system performance.

Thus, using the Dhrystone benchmark, we can project the performance of each CPU in a Balance system to be about 1.4 times the performance of the VAX 11/750 for a single-stream, CPU-bound integer application, or approximately 0.8 MIPS.

More importantly, the performance of the system is linearly related to the cumulative sum of each individual processor's performance. Using multiple copies of the test as an approximation of a multiprocessing benchmark, we see that the throughput of the Balance system grows almost linearly with the number of processors. When 12 copies of the test are run on 12 processors, the cumulative throughput is an average of 11.5 times the throughput of a single processor. It may be concluded that the Balance 8000 configured with 12 processors has the CPU power of a dozen VAX 11/750's for representative applications. The cumulative performance number is only useful when considering programs with little or no dynamic sharing—as in a multiuser application, for example.

As another measure of system performance, the multiuser benchmarks show that a fully configured Balance 8000 with up to 12 processors can support up to 96 users, and a Balance 21000 system with up to 30 processors can support up to 256 users.[12]

The second measure of performance is the speedup of a single application program using multiple processors as compared with a single processor. It is clear that the application speedup cannot exceed the effective compute power, but it can be less. The application speedup can be degraded by a number of factors. An application may not have enough parallelism to use all of the available processor cycles, it may suffer from application-level contention, or it may be limited by other, unconsidered factors. Thus it is difficult to generalize the performance results for any one application. In one application, parallelization of a standard floating-point program, Linpack, resulted in improvement by a factor of 27.4 with a 30-processor system.

We also studied the Balance 8000 cache and bus protocols.[13] The single-thread cache performance shows that the cache performs well in integer applications (95-percent hit rate). This rating results from the high locality in the applications and the 8-byte line size, which permits implicit prefetching of instructions and 32-bit data. Performance degrades in double-precision floating-point applications since accessing 64-bit-word data breaks the implicit prefetching strategy (85-percent hit rate). For multithread applications, cache performance across the systems depends on load balancing. Performance is generally superior to the single-thread version of that application, because the data has been distributed and because of the larger total cache space.

For the parallel application, the number of writes is the limiting factor for multiprocessor systems with write-through caches, rather than the sharing between the processes. The writes from each processor can swamp the bus in a large configuration of such a multiprocessor system. A large amount of write invalidations also increases the cache miss rate for parallel applications. Write invalidation at the cache indicates sharing among the processors. Study results show that the write invalidation for observed parallel applications is indeed very small and thus indicates little sharing. More studies will be needed to observe the behavior of these applications with larger numbers of processors than those used in this experiment.

The study of bus utilization for multithread applications acknowledged the large proportion of idle time occurring in the bus (under 25-percent utilization for an eight-processor system). However, increasing the number of processors in the Balance system beyond 30 would degrade the bus performance due to the number of writes occurring on the bus. The multiuser benchmark showed that (1) the bus was less of a limiting factor, and (2) potentially the number of processors can be extended beyond 30 with a copy-back cache policy provided that I/O capability is also extended. Each parallel application has different types of bus utilization characteristics. Some indicated large amounts of write traffic and others didn't.

The performance of Balance in the multiuser environment is significantly superior to a uniprocessor system, mainly because the independent processes show little sharing. This translates to a lower cache miss rate. Thus a Balance type of multiprocessor seems to be a natural fit for the multiuser and some parallel-application environments.

Multiprocessing, a powerful technology, provides a means to linearly amplify the power and economy of VLSI microprocessors. Today, the technology has advanced to the point that it is possible to build a scalable, low-cost, high-performance multiprocessor on which many applications can run. Some applications can be speeded up with little or no work, and the first tools are in place to selectively adapt those applications that need restructuring. Accelerators may be developed to provide quantum performance improvements for specific applications.

We expect the next-generation multiprocessors to offer increased power and a set of applications that see major benefit from the continued growth of multiprocessing. Compilers that automatically recognize parallelism should evolve, designers should continue to develop better parallel algorithms, and debuggers and languages that allow the expression of parallelism should become more prominent.

Acknowledgments

We thank Bob Beck and Bob Kasten for their contributions to this article and the rest of the people involved in the making of Balance.

References

1. G. Fielland and D. Rodgers, "32-bit Computer System Shares Load Equally Among up to Processors," *Electronic Design*, Sept. 1984.
2. National Semiconductor Corporation, *The 32000 Family Databook*, Santa Clara, Calif., 1986.
3. G. Efland and W. Mayberry, "Cache Boosts Multiprocessor Performance," *Computer Design*, Nov. 1984.
4. A.J. Smith, "Cache Memories," *ACM Computing Surveys*, Sept. 1982.
5. B. Beck, B. Kasten, and S. Thakkar, "VLSI Assist for a Multiprocessor," *ACM/IEEE Proc. ASPLOS II Conf.*, Oct. 1987, pp. 10-20.
6. F.J. Corbato, "Paging Experiment with the MULTICS System," MIT Project MAC-M-384, Cambridge, Mass., May 1968.
7. P.J. Denning, "Virtual Memory," *Computing Surveys*, Sept. 1970.
8. H.M. Levy and P.H. Lipman, "Virtual Memory Management in the VAX/VMS Operating System," *Computer*, Mar. 1982.
9. Sequent Computer Systems, Inc., *Balance 8000 Technical Summary*, System Manual, Beaverton, Ore., 1985.
10. D.P. Rodgers, I.L. Johnstone, and S.L. Baldrige, "Multiprocessing in a Software Development Environment: Parallel Tools," *Proc. Wescon*, 1985.
11. R. Weicker, "The Dhrystone," *Comm. ACM*, Oct. 1984.
12. Sequent Computer Systems, Inc., "Multiuser Workload Performance," Tech. Note, Jan. 1986.
13. S. Thakkar, "Performance of Shared-Memory Multiprocessor Systems," *Proc. ICCD 87*, Oct. 1987, p. 63.



Shreekant Thakkar is senior systems engineer at Sequent Computer Systems. His research interests include highly parallel systems, computer architecture, parallel programming and algorithms, and VLSI design.

He received his BSc in statistics and computer science from the University of London and his MSc and PhD in computer science from the University of Manchester. He is a chartered engineer and a member of the IEEE, ACM, British Computer Society, Britain's Institute of Electrical Engineers, and the editorial board of *IEEE Software*.

Paul Gifford is manager of Sequent's central systems engineering and the program manager for Symmetry platform developments. His current interests include computing system architectures, processor and cache designs, system interconnect technologies, design automation tools, performance evaluation techniques, and system platform developments.

Gifford received his MSEE and BSEE from the Rochester Institute of Technology. He is a member of the Computer Society of the IEEE.



Gary Fielland is director of advanced development and a cofounder of Sequent Computer Systems, Inc. He was responsible for the architecture of the Parallel Programming Environment family of computers, the SLIC subsystem, and the current parallel software offering. Currently, he is architect for the ongoing development of future systems products.

Fielland holds a master of engineering degree in computer science and electrical engineering from the University of Florida and is a member of the IEEE, ACM, and SIAM.

Questions about this article can be directed to Shreekant Thakkar at Sequent Computer Systems Inc., 15450 SW Koll Parkway, Beaverton, OR 97006-6063.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

Low 156 Medium 157 High 158

MicroLaw

Richard H. Stern
Law Offices of Richard H. Stern
2101 L Street NW, Suite 800
Washington, DC 20037

Reflections on Dirty Harry and dBase

Ed Esber, CEO of Ashton-Tate, has threatened to sue the sellers of "add-on" programs that work with dBase.[1]

Speaking at the annual meeting of the Software Publishers Association, Esber stated that he opposed the efforts of a committee of add-on firms seeking to standardize the dBase programming language. He threatened to sue them for copyright infringement, antitrust violations, and trademark infringement. He concluded his remarks with this invitation to the committee to persist in their project: "Go ahead—make my day."

Esber here invoked the spirit of Dirty Harry, the celebrated Clint Eastwood cop. Esber's invitation paralleled that of Dirty Harry to a store-heisting punk. Harry dared him to reach for his gun so that Harry could blow him away with his Magnum. The punk blinked. Should the dBase standardization committee blink too?

Fox (the publisher of FoxBase), SBT, and Wallsoft Systems do not plan to blink—yet. Instead, they sought IEEE microprocessor technical committee support for establishing a dBase programming language standard.[2] In response, Ashton-Tate reportedly reasserted that their copyright on dBase programs covers the dBase language and that A-T will enforce its rights against infringers.

Whether the standardizers ought to blink probably depends on whether Esber's threats pack any real punch or are just so much hot air. Does A-T have any copyright monopoly over dBase? Would the small add-on firms violate the US antitrust laws by beating up on giant A-T (when they agree how to write their computer programs in dBase)? Will they commit trademark infringement if they dare to label their standard something like "add-on developer's standard format for writing computer programs in dBase programming language?" Or label their products similarly?

Does A-T have any legitimate gripe about anything at all? If it does, can it take effective action by filing copyright, antitrust, or trademark lawsuits—or by any other means?

"Go ahead—make my day."

Finally, what are the implications for the IEEE and the Computer Society? Both have reputations for their commitment to standardization. Is a copyright, antitrust, or trademark Magnum ready to zero in on IEEE and Computer Society standardizers and blow them away? Will an IEEE standard for dBase be illegal? And what about the ANSI/ISO draft C standard? Are the ANSI standardizers lawbreakers too?

Copyright infringement

Ashton-Tate holds a copyright on its user manuals and the code in the diskettes furnished to its dBase customers. That copyright means that nobody can copy the manual or diskettes with impunity, unless A-T gives its permission. The copyright does not prohibit anyone from writing a book about dBase. Neither does it prohibit programmers from writing any code they want to create in the dBase programming language.

One hundred years ago the Supreme Court decided that if you publish and copyright a book about how to do double-entry bookkeeping, to make watches, or to cure diseases, nobody may copy your book and sell it. But anybody who wants to may use the teachings of your book to perform bookkeeping, to manufacture watches, or to cure sick people.

A copyright is not a patent. It would be "a fraud upon the public," the Supreme Court said, if a copyright were treated like a patent. The teachings of a copyrighted book belong to the public when the book is published. That is one purpose of publishing books and of the US copyright laws. If parts of the book, such as blank forms, are vital to the teachings of the book, they also fall into the public domain as a "necessary incident" of the teachings. Whether it makes sense or not (much of the time it doesn't), the US copyright law treats computer programs as if they were books. Everything that the Supreme Court said about copyrighted books on double-entry bookkeeping a hundred years ago applies to copyrighted computer programs today.

Good or bad, that means that when A-T published its user manuals about dBase I, II, and III, and taught the world how to speak dBase, it dedicated the dBase programming language to the public. So did Nicolas Wirth when he wrote his book about Modula II, as did

Ritchie and Kernighan when they wrote their book on C. Even in the absence of the books, the writers of those languages would have no copyright monopoly over the languages by virtue of having created them. The US Copyright Act does not protect every conceivable aspect of everything that somebody creates and writes down on a sheet of paper. The Copyright Act expressly states that no "idea, plan, procedure, process, system, method of operation, concept, principle, or discovery" can be protected under the US copyright law. Languages fit within this provision, and therefore they are not the kinds of things that copyright law protects. Like algorithms and flowcharts, you can create languages, but then anybody can use them without paying you one cent for the privilege.

Therefore, when one of the add-on developers, like Fox or SBT, writes programs in the dBase programming language, A-T cannot sue them for copyright infringement and expect to prevail. The add-on firms can even write their programs in a mutilated, derivative version of dBase (as A-T might see it), and it still would not be a copyright infringement. The end users who buy Fox's or SBT's software are also free from copyright liability when they use the software. Nobody in the chain of distribution is liable for copyright infringement. The copyright laws are no Magnum against "unauthorized" use of a programming language.

Antitrust violations

Now to Esber's charge of antitrust violation, based on the allegation that this gang of Davids is beating up on Goliath. Ashton-Tate will sue the add-on gang, Esber warned, if they "act in concert to control the dBase programming language." First, how are they going to control the dBase programming language? Second, even if they achieved the impossible and controlled the dBase programming language, how will that monopolize or restrain trade in any relevant market[3] in violation of an antitrust law?

Is dBase a relevant market? Would A-T really want to allege that? By alleging it, A-T might admit that the same is true in other contexts, such as when a dealer that A-T cans from its distribution channel counters with an antitrust suit against A-T (as the cutoff Apple dealers did against Apple).

Is there something inherently evil or immoral about setting a standard—so that it ought to be an antitrust matter regardless of the economic facts? No, standards generally benefit the public because they help everybody transfer their knowledge from one application to another. Standards help individual end users learn faster and avoid confusion. They stem the exasperation caused by inconsistent user interfaces when you can't remember which one to use. Standards help corporate users because they permit employees to transfer their knowledge from one program to another. Training time decreases, costly mistakes diminish, and worker productivity increases. Standards even help programmers avoid reinventing the wheel, saving time and effort. In short, standards usually serve the public good.

The copyright laws are no Magnum against "unauthorized" use of a programming language.

The antitrust laws have no bias against legitimate standardization efforts. When I was chief of the Antitrust Division's Intellectual Property Section in the 1970's, I would have laughed anybody out of my office who wanted me to sue some industry group for trying to standardize a computer programming language. Nowadays, in a more relaxed antitrust environment, can you seriously imagine an antitrust action against these small add-on firms for allegedly ganging up on a Goliath by setting standards? Dominant firms in an industry (perhaps acting through a trade or professional association) can prevent competition from new technologies by establishing standards to bar any new technology from the marketplace. But that is a far cry from this kind of situation. The antitrust Magnum here is not even a popgun.

Trademark infringement

Under US trademark law, you must not use somebody else's trademark to create confusion about who sold or sponsored the product. Fox or SBT must not imply to customers that A-T approves, sponsors, or is in some way responsible for the dialect of dBase that Fox or SBT uses in its add-on computer programs. That implication could cause customers to become angry at A-T if they disliked the Fox or SBT program. Fox and SBT should not control what happens to A-T's reputation in the marketplace. The same principle would apply to an IEEE or ANSI standard for dBase, 1-2-3, or Unix.

On the other hand, "dBase" is not a taboo word. It is perfectly legal for add-on sellers to make truthful comments about their programs, such as "This program works with dBase III," or "We wrote this program in the Software Publishers Club Standard Version of dBase language." Courts have repeatedly ruled that a company could advertise *truthfully* that its product was compatible or worked with some other named product. At most, the law would require the add-ons to use a notice that "Ashton-Tate did not approve or sponsor the version of the dBase programming language used in writing this program." They might refer to dBase as "D-base."

Furthermore, simply meeting to set standards is not a conceivable basis for a trademark infringement action. Before trademark infringement can occur, someone has to market the goods using the trademark said to be infringed. How the add-ons marketed their products would determine the infringement, not what they did at their standards meetings. Of course, the same principle applies to the IEEE, which ordinarily does not market goods.

The trademark Magnum, also, is not loaded. Nobody is going to make Esber's day, after all—not under the copyright, antitrust, or trademark laws, anyway. The add-on companies, and by the same token the IEEE, can go on trying to formulate standards and nobody is going to blow them away. So much for the first part of these reflections on Dirty Harry and dBase.

Does Ashton-Tate have any legitimate grievance?

The press account doesn't really say what A-T is mad about, beyond the principle of the thing. Perhaps it is just the principle. (It's my programming language: Nobody else should mess around with it!) Maybe it's the idea that one thing leads to another, and there you are careening down the slippery slope. After enough kicking around, there would not be much of a trademark left.

dBase might suffer "genericide," as did the trademarks, "aspirin," "escalator," and "thermos" (all once-valid trademarks, now generic names).

Maybe the add-ons could end up dictating to A-T how to write dBase or what to call commands. Perhaps their standardization efforts would so capture the minds of end users that A-T would be forced to do things like call the command Satisfy something else, like Valid. Maybe the standards makers would deny A-T the option of adding new extensions. Not very likely, but it might look scary to a timid CEO.

Actually, a potential legitimate grievance does exist. In its blustering, A-T has left this point unstated. The US copyright laws (and other industrial property or intellectual property laws) do not provide any kind of protection for creating computer programming languages. Somebody like Nicolas Wirth spends thousands of hours writing a language that improves the world; other people make money writing and selling computer programs in that language; but none of them gives Wirth a dime for his trouble. Somebody like A-T pays programmers thousands or hundreds of thousands of dollars to write the dBase IV program and its language; other people make money selling programs written in the language; and none of them makes charitable contributions to A-T.

Whether your are moved to weep for Wirth or A-T, you can see that incentives for creating new languages are diminishing to meet the limited rewards. In Wirth's case, I suppose he got nothing but an enhanced reputation. A-T made whatever it made (millions of dollars) by selling copies of the dBase I-III programs. (Of course, considerably more went into all of that besides writing the dBase language. They conducted market research to figure out what product features or task solutions users would want to buy, designed the screens, wrote the program, and created the documentation. A-T made millions of dollars because of all those other things, too. The compensation for creating the dBase language itself just sort of got folded into everything else or was ignored.)

Sometimes, the system works. After all, we do have dBase and Modula II. We have C, Pascal, Basic, and hundreds more. Of all of them, only a few are big sellers. Do we need more? Do we want to have some mechanism for encouraging more languages to be written? Would that further software progress, the expansion of the computer and computer software markets? We have a mechanism for rewarding and thus encouraging people who can bundle a new programming language in with a marketable software product. But we do not have a mechanism for rewarding and thus encouraging people simply to write new programming languages. That is probably a mistake.

The idea of having such a mechanism does not require this mechanism to reside in present US copyright law. Other (possibly better) ways to encourage creativity and investment in technology exist. The Semiconductor Chip Protection Act, for example, gives limited, 10-year, noncopyright protection for chip layouts. Copyright law's 75 years of injunctions, punitive damages, and prison sentences are not necessary to encourage creation. Somebody like Wirth would probably be quite happy to receive a very modest royalty; that prospect might mobilize scores of writers of future great programming languages. The assurance of a modest royalty from add-on sellers might even encourage A-T to put forth better enhancements of the dBase language (and pay the salaries of employees to write them). That would be a better system than we now have.

If you put aside all the hot air in the "Go ahead—make my day" speech, a core of presently unredressable, legitimate grievance remains. A-T cannot use the copyright, antitrust, or trademark laws to suppress standard-setting by the add-ons or the IEEE. End users would benefit from a standard type of add-on. But the way our present intellectual property law works, add-on publishers are getting a free ride on dBase and on A-T's development costs. It would be wrong to stop the add-ons from using dBase, or C, or any other language. But perhaps the law should require 50 cents or a dollar or two per diskette to compensate the creators.

Would an enhanced intellectual property system better serve software users and publishers? Languages, moreover, are only the tip of the iceberg. There is the whole screen display controversy, recently aired at US Copyright Office hearings.[4] There are issues about sets of keystrokes (such as the controversy over who, if anybody, owns </FR> for spreadsheets) and other user interface disputes. There are struggles over icons, metaphors, and instruction sets. There are also people out there who think they ought to "own" computer program specifications. Somebody just talked a federal court in Atlanta into believing that they own the use of high-intensity video (highlighting) and capital letters in menus, at least for command-driven menus for modem software.[5] While some of these claims seem outrageous in the context of 75 years of copyright injunctions, massive damages, threats of imprisonment, and so on, if such a claim were scaled down to 50 cents a diskette, it might not seem so unreasonable.

Actually, the system may be changing before our eyes: Some of these claims are succeeding and some of these threats of suit are scaring off start-ups. (What about the first GEM interface?) But little or no public debate is occurring in the industry about what would make sense here. Certainly no coherent evaluation and balancing of interests is taking place as Congress might do. What A-T threatens about dBase standards is probably too far from the mainstream to work in court, but plenty of other claims of the same quality are being pursued and may succeed either in court or in the boardroom. Is Esber's speech a signal that we need to talk this over? Could we work out a compromise?

References

1. "Ashton-Tate's Esber warns dBase language committee," *C&SN*, Oct. 19, 1987, p. 1.
2. *InfoWorld*, Dec. 7, 1987, p. 8.
3. "Relevant market" is a term used in antitrust cases to refer to a cluster of products or services that are sufficiently similar to and substitutable for one another that they constitute an economic unit. It is meaningful to speak of monopolizing something only when that something is a relevant market. It does no good to gain a monopoly over product A if users can readily turn to products B through N.
4. MicroNews, *IEEE Micro*, Dec. 1987, p. 90.
5. MicroLaw, *IEEE Micro*, Dec. 1986, p.77; June 1987, p. 81.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 171 Medium 172 High 173

MicroStandards

Michael Smolin
Smolin & Associates
3428 Greer Road
Palo Alto, CA 94303

IEEE Standards Board actions

At its most recent meeting in December 1987, the IEEE Standards Board took some actions of interest to us. The Board withdrew project P856, Methods for Evaluating Microprocessor Performance. No one responded to my plea in last issue's MicroStandards for interested parties to come forth to maintain this project. If any of you are interested in restarting P856 (informally known as "Microprocessor Benchmarks"), please contact me.

New approved standards

At its recent meeting, the IEEE Standards Board adopted as standards the drafts created by two projects of the Technical Committee on Microprocessors and Microcomputers (TCMM). These standards are now known as:

- IEEE Standard 961, Eight-Bit Microcomputer System Bus (STD Bus); and
- IEEE Standard 1000, STE Bus.

Publication of new standards

Recently published IEEE standards include *ANSI/IEEE Std. 854-1987, Radix-Independent Floating-Point Arithmetic.* The IEEE approved it in March 1987, ANSI approved it in September, and the IEEE published the standard in October. You may order this standard by phone, (201) 981-0060. Ask for item SH11460. The price is $17.

You may also want a copy of *ANSI/IEEE Std. 610.2-1987, Glossary of Computer Applications Terminology.* This is item SH11064 and costs $11.50.

Sponsor ballots

Three sponsor ballots are now completing. These ballots are to recommend the reaffirmation of *IEEE Standard 796, Microcomputer System Bus* (Microbus I) and to recommend the adoptions of P1096, Multiplexed High-Performance Bus Structure (VSB), and P959, I/O Extension Bus (SBX), as IEEE Standards. They are expected to pass and could go to the IEEE Standards Board at its March 1988 meeting.

In the queue for sponsor ballots are P970, Advanced Backplane Bus (Versabus), and P1132, VMS Bus. P855, revision of *IEEE Std. 855, MOSI-Standard for Microcomputer Operating System Interfaces* (Trial Use), will soon join them.

Drop me a note if you want to participate in any of the sponsor ballots.

The sponsor

The sponsor for most of the microcomputer/microprocessor/microcontroller-oriented standards is the TCMM. Its standards subcommittee is the Microprocessor Standards Committee, which oversees the working groups for the sponsor. The MSC meets in the evenings on the second Monday of odd-numbered months in Los Gatos, California. Call MSC Chairman Clyde Camp, (214) 995-0407, for agenda or other information. Those interested in starting a project should reserve agenda time to make a presentation on their subject.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 174 Medium 175 High 176

MicroReview

Richard Mateosian
2919 Forest Avenue
Berkeley, CA 94705
(415) 540-7745

Afterthoughts

This is my sixth MicroReview column, hence the end of my first year at this post. In my first column, I asked you readers to use the editorial response cards to let me know what reviews would interest you and what you thought about what I had written. So far, no takers; so I'll just keep doing what I've been doing—reviewing books and software that interest me. Still, a little feedback would be nice.

In June 1987 I reviewed a book on CD ROM for optical publishing. Since then, Bowker Electronic Publishing has deluged me with press releases. In addition to CD ROM versions of their standard *Books in Print* and variants thereof, they have now released a CD ROM version of the *Oxford English Dictionary*. Two 4.72-inch disks contain the entire 12-volume dictionary and eight indexes. I can't tell from the press release what hardware you'll need to use this electronic OED, but the two disks cost $1250.

In August 1987 I told you that microprocessor interfacing techniques is the "hot" subject in computer books, and I reviewed a very good book on that subject. This month I've got another one (read on), and it's also good.

In December 1987 I wrote an enthusiastically favorable review of Microsoft Word, version 3.01. Since then I've used Word exclusively for all my word processing, so I must still have a good opinion of it. But over time a dark side has also appeared. For one thing, press reports have repeatedly noted bugs. I have noticed peculiar and "unintuitive" behavior from time to time, but nothing that would cause damage. It's hard to say how much of this behavior is the result of bugs, since the Microsoft manuals are often not very specific about how Word's features should work.

A more serious problem is that Microsoft did not thoroughly design some of Word's features. The styles feature is a good example. The basic idea is simple enough. Suppose that I use the 12-point New York font for most of my document and wish to set quoted paragraphs in 10-point Times, indented 0.5 inch from current left and right margins. All I need to do is set up one quoted paragraph as described, select it, and then use the Define Styles command to assign a name—say QuotPar—to the style. Whenever I select any other paragraph and use the Styles command to assign the QuotPar style to it, the new paragraph is then formatted like the first one.

So far this is good, but problems arise when things change. Word will internally record the QuotPar style defined above, for example, as something like:

Normal + (Font = Times) + (Size = 10 point) + (Left indent = 0.5 inch) + (Right indent = 0.5 inch)

The initial term "Normal" refers to my base style, which means (for this example) 12-point New York with no left or right indentation. All the other terms are absolute replacements for the corresponding parameters of Normal. Thus, if I change Normal to a 24-point Helvetica font, the quoted paragraphs will still be 10-point Times. If I change Normal to have left and right indentations of 1.5 inches, my quoted paragraphs will retain their 0.5-inch indentations and intrude an inch into the margins on either side.

Other style elements follow ad hoc rules. For example, if QuotPar calls for italic type, it will render existing italics into roman type when it's applied to the new paragraph.

The problem is not necessarily that Microsoft has chosen its rules illogically, but that the user has no control. The user's intuitive expectation of dependent style changes based on parent style changes is often at odds with the way Word behaves. Reducing a full-width document to a narrow column format is one example. Naturally, an intricately formatted full-page table cannot be changed conveniently to fit into a narrow column, but I would expect Word to reformat QuotPar paragraphs automatically to fit the narrower space.

Much of Word's unintuitive behavior concerns the way styles relate to column width, and some of this confusion seems to carry over to programs like Pagemaker, which import files produced by

Word. I have no idea whether the problems that I've observed with such file transfers are Microsoft's fault, Aldus's fault, or nobody's fault; so I won't go into them further here.

Odds and Ends

Fast Access/WordPerfect, Rhyder McClure, Prentice-Hall Brady Books (New York, 1987, 200 pp., $12.95)

What made me notice this book was the quotation on the cover:

> Rhyder McClure has written the first book about computer software that does not appear to have been translated from the Japanese.
> —Andy Rooney

The remark is patently untrue and has nasty undertones. As Andy Rooney probably knows, the readability of any English translation depends almost entirely on the ability of the translator to write English. The relative unavailability of native speakers of English who understand Japanese leads to the hilarious instructions that sometimes accompany items produced by Japanese firms. The situation that Andy Rooney mocks, to the delight of Prentice-Hall's marketeers, does not result from an inability of Japanese speakers to express themselves. It springs from a general unwillingness of Americans to learn the language of a people whose products they crave insatiably.

Rooney's remark is only tangentially aimed at Japanese manuals. Its main implication is that Andy Rooney has had difficulty understanding all books about computer software, up until this shining example. This may be true. In fact, McClure's book—while clearly written—is pitched at a pretty low level.

McClure sets out to provide a step-by-step guide that is more detailed than WordPerfect's on-line help and clearer than its manual. However, he covers only those features and commands of the WordPerfect word processor that "95 percent of us use 95 percent of the time." I don't know much about WordPerfect, but I think that McClure is overestimating the percentages. In any event, a book about Microsoft Word covering the material that McClure covers in this book would be valuable to only the most inexperienced users. It would be no use at all to those of us who spend 95 percent of our time trying to figure out how to use the other five percent of the commands.

Finally, let's look at two examples of McClure's exemplary English, both taken from the first 10 pages of the book:

> The system treats the <**ENTER**> Key no differently than any other..;.tap the **Backspace Key** and the cursor drops to it's former position (the Hard Return, as a tap of the <ENTER> Key is known in WordPerfect is deleted).

> Cursor to the file you want to work on and tap 1 (1 is **Retrieve** from the choices at the bottom of the Directory. Don't use the letter **l** for the number **1**).

Microcomputer Buses and Links, D. Del Corso, H. Kirrmann, and J.D. Nicoud, Academic Press (London and Orlando, Fla., 1986, 415 pp., $25 and $45)

(*Disclaimer: Del Corso is currently a member of* IEEE Micro*'s editorial board, as was Nicoud for a number of years. We generally have little contact with one another, and in any event this association had no effect on my decision to review the book or on the content of the review.*)

I've always liked the way Academic Press does technical books. I wish they would send me more of them to review. This one is nicely put together with many extremely clear illustrations. My only complaint about its appearance is the slightly odd typeface they used.

The authors provide a clear, well-organized exposition of the concepts underlying bus design. They hope to provide design engineers with the tools they need to select and work with existing standard buses. The appendix (35 pages of small print) presents, in a consistent format, information about the hardware associated with a selection of standard serial and parallel links and backplane buses.

The authors claim that engineers with a minimal knowledge of the field can read this book, and this is certainly true of the parts that I read. (I have a minimal knowledge of, but a great interest in, the field.)

This book is the kind that's hard to resist in a bookstore but a lot of work to read. Readers are expected to follow the structured development of the material through 10 chapters; however, a short index leads readers directly to major references to specific points. The book is obviously suited for use in an academic setting, but the authors give no recommendation for the type, level, or duration of instruction.

Suitcase and *Power Station*, Software Supply (Sunnyvale, Calif., $59.95 each)

Software Supply has given us two well-designed programs for the Macintosh. Suitcase allows you to keep fonts and desk accessories in separate files, rather than bundled with the system file or with specific applications. As a program run at start-up time, Suitcase automatically opens fonts and desk accessories in certain folders in the system file. As a desk accessory, it allows attachment of fonts or desk accessories to an application that is already running. In other words, Software Supply has corrected an irksome flaw in the Macintosh design by introducing modularity into the handling of fonts and desk accessories.

Power Station provides convenient launching of applications and desk accessories. A Power Station screen is a grid of 27 positions. Each grid position can be assigned to an application, a desk accessory, an application with one specific document, or an application with a window. The names of specific documents can be placed into the window for easy selection. Each grid position initially displays the name of the item assigned to it, but you can rename it. When you launch Power Station—perhaps as the start-up application—the home screen appears, and you can quickly access 15 other screens from there.

The purpose of Power Station is to cut across the hierarchical Macintosh file structure, bringing the selected applications and documents together, regardless of their positions in file folders. This function leads to a few problems synchronizing the Power Station screens with changes produced by the Finder or from applications. Some of these problems could be alleviated if entire folders, rather than just specific documents, could be associated with applications installed in Power Station grid positions.

Since receiving these programs, I've integrated them into my system and have become quite dependent on them. I recommend them highly.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 177 Medium 178 High 179

MicroNews

MicroNews features information of interest to professionals in the microcomputer/microprocessor industry. Send information for inclusion in MicroNews one month before cover date to Managing Editor, IEEE Micro, *10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578.*

Superconductivity update: Who, What, and How

IEEE Micro reports on materials and technology, research to evaluate sources of raw materials, and some of the challenges associated with developing superconductors.

Researchers debug ceramics

Problem: high resistance occurs where external leads are attached to ceramic superconductors. Reason: exposure to air causes the surfaces of ceramic materials to degrade. Some two-month-old samples are 10 times more resistant than fresh samples.

To combat this problem, Westinghouse Research and Development Center and the National Bureau of Standards produced a new method of making low-resistance electrical contacts on ceramic superconductors. They developed a three-part process: (a) minimize air exposure time, (b) sputter etch the surface to remove the degraded layer, and (c) deposit a thin layer of noble metal (silver or gold) to protect the ceramic surface from degradation.

Researchers etched bulk yttrium barium copper oxide to a depth of 20 to 50 nanometers at 1.25 kV rms in 3-Pa argon for three minutes. They immediately sputtered a 1- to 6-micrometer-thick contact pad onto the surface (4.2 kV rms, no applied bias), also under argon atmosphere. During the process, a water-cooled sample holder kept the superconductor temperature to less than 100° Celsius.

Wafers cook, crystals grow

Are you looking for samples of superconductive material to study? Then you might want to join the line at The Bakery, a small lab at Texas A&M run by R.K. Pandey. The Bakery cooks up quarter-size, wafer-like samples of yttrium barium copper oxide for physicists, chemists, and engineers studying its properties.

On the other hand, if you're seeking ferroelectric single crystals, Pandey grows them up to 6 millimeters in size.

According to Pandey, single crystals are candidates for use in electro-optical devices and as substrates of superconductive material. Researchers are working to improve materials for film and single crystals to use in computer memories, logic, data transmission, and magnetic field and infrared detectors. They hope to introduce the speed of superconductivity into these functions.

Pandey feels the single crystal is extremely important because it will allow scientists to understand what really causes superconductivity in this material. Single crystals can be much better understood than polycrystals because they do not have the same kinds of defects.

Pandey offers to work "on a short-term basis" with representatives of science and technology and will sell samples, with cost dependent upon need. "We produce single crystals on a regular basis for our own research, but we can speed up production for those who need these crystals," he said.

Contact R.K. Pandey at Texas A&M University, Department of Electrical Engineering, College Station, TX 77843.

Where's the beef?

When the technology comes around, will there be enough material to go around?

Falmouth Associates, a technical consulting firm, is investigating that question with a six-month study of superconductive starting materials. The study measures the world's current supply of raw materials and evaluates whether present separation and refining techniques can meet the commercial demands expected within the next five years.

The report details how materials are refined and prepared for use, describes and assesses the fabrication routes to superconductors, and forecasts the demand for these materials by specific end uses.

End-use forecasts include both thin-film applications (microcircuit elements, SQUIDs, and infrared detectors) and bulk applications (wires, magnets, motors, and power transmissions).

Subscribers obtain a complete business analysis and commercial information on suppliers, fabricators, and pricing and trade trends. A section of the report deals with new business opportunities in the field of semiconductors.

For further information, contact Falmouth Associates, Inc., 170 US Route One, Falmouth, ME 04105; (207) 781-3632.

Heads up!

In a related event, US National Bureau of Standards Director Ernest Ambler has warned that vital changes in policy must occur to promote the exploitation of superconductivity.

At a November 16 conference to assess the commercial realities of this new field, Ambler stated that both corporations and government must work to lower the barriers that block commercial development of superconductivity and other emerging technologies. He cited inadequate long-range planning, overemphasis on product versus process technology, and improper integration of research, development, manufacturing, and marketing. Ambler advised listeners to learn from past mistakes.

MicroTidbits

The American Telephone & Telegraph Co. Foundation awarded research grants totaling $60,000 to the **University of Michigan's College of Engineering**. The fields of signal processing, optical communications, and integrated circuits shared the grant equally. AT&T awarded $3 million to schools in 1987 to enhance the education of future technical leaders.

CompuServe plans a March 1 pilot program in 10 major US cities to provide 9600-bps dial-up access for commercial customers. According to the company, the Hayes V-Series Smartmodem 9600 provides significant error control and adaptive data compression capabilities.

Production of **Toshiba America** laptops reached 10,000 units per month at its Irvine, California, plant recently. Toshiba manufactures the T1100-Plus, the T1200, the AT-powered T3100, and the 80386 microprocessor-based T5100 at the Irvine facility, which employs 500 people.

MIT contracted with **Nynex Corporation** to conduct collaborative research in five of the school's major laboratories. The research focuses on eliminating certain switching functions in communications networks, developing an intelligent computer program to support software development, and improving the efficiency of quality software production. MIT provides academic insights for potential products and services; Nynex stimulates interdisciplinary investigation.

The Electronic Publishing Lab at the **National Bureau of Standards** officially opened. The lab promotes understanding of technologies underlying electronic publishing, furthers development of standards and guidelines, and showcases desktop publishing systems for government agency evaluation. Twenty-four companies have either donated or loaned software and equipment.

The 80386 outruns the 68020

According to Forth, Inc., the Intel 80386 performed 25 to 40 percent faster than the Motorola 68020 in recent polyForth 32-bit benchmark tests. The 80386, heart of IBM's Model 80 PS/2, surprised Forth by switching front-running positions with the 68020, core of Apple's Macintosh II.

Forth explains: "Our polyForth operating system was one of the first to run on the 80386 in full 32-bit mode. The other benchmarks were [conducted] on operating systems running the 80386 in compatibility mode."

The 80386 used an Intel 386/21 Multibus SBC, while the 68020 employed a Mizar 7120 VMEbus board. Both ran at 16 MHz and possessed cache memory.

Forth also benchmarked the NCR-32 running in an NCR-9300 computer at 6.7 MHz. The NCR-32 uses a microcode implementation of polyForth.

DEC's MicroVAX II and Intel's 16-bit 80286 completed the sampling. The VAX II ran a native polyForth system without DEC software. Forth also ran other comparison tests using 16-bit math on the 80286 at 10 MHz on an IBM PC AT. (See Table 1.)

The compatibility of polyForth with all five processors allowed direct comparisons with benchmarks written in a high-level language. (polyForth is both a language and its own operating system.)

For further explanation of test procedures, contact Forth, Inc., 111 N. Sepulveda Blvd., Manhattan Beach, CA 90266; (213) 372-8493.

Table 1.
32-bit processor benchmarks.

Test #	Tests performed/ms	80386 16 MHz, cache	68020 16 MHz, cache	NCR-32 6.7 MHz, microcode	VAX II native	80286* 10 MHz, 16 bits
1	100-K empty loops	282	392	285	652	769
2	10-K multiply/divides	156	244	832	434	384
3	10-K multiplies	109	144	349	264	329
4	10-K divides	125	180	411	300	329
5	10-K unsigned divides	125	176	400	352	274
6	Sieve, optimized, per cycle	221	372	305	720	879
7	String search (*EDN* E), μs	66	228	128	320	164
8	Context switch, μs	9.9	9.6	8.8	22.4	27.4
9	Complete Sysgen, s	6.708	8.7	7.11	11.89	12.967

* Tests 2 through 5 used 16-bit math on the 80286.
(Source: Force, Inc.)

IEEE Software Call for Papers

IEEE Software, a bimonthly publication of the Computer Society, seeks articles for its November 1988 issue on **expert-system software.** Contributions should concentrate on systems that help program developers enforce standards, suggest alternative methods, and help create documentation. **Deadline: April 1, 1988.**

Also invited are articles on human-computer interfaces, software technology in the Far East, and rapid prototyping for 1989 special issues and other software topics for general-interest issues in 1988 and 1989.

Submit eight double-spaced copies of completed manuscripts to Ted Lewis, Editor-in-Chief, *IEEE Software*, c/o Computer Science Dept., Oregon State University, Corvallis, OR 97331; (503) 754-2744; CSnet lewis@oregon-state.

COS announces test service

The Corporation for Open Systems offers a 8802/4 Physical and Data Link Layers Testing and Consulting Service at its headquarters in McLean, Virginia.

This service tests protocol implementations against the current version of the International Organization for Standardization 8802/4 standard. The ISO standard specifies electrical signaling, frame formats, actions of the stations upon receipt of a data frame, and station management functions.

The test service is conducted on two layers. The Physical layer is based primarily on Hewlett-Packard instruments and controllers. The Medium Access Control layer is based on Vance System's CTS 40 conformance test system. Tests use COS-developed, executable test cases based on the generic test purposes proposed by the IEEE 802.4-J working groups.

COS also promotes international open systems interconnection/ISDN standards. The Standards Promotion and Application Group (SPAG) Services of Brussels, Belgium, and COS plan to harmonize functional profiles, test tools, and test suites.

The start of something big?

A recent Edinburgh University conference found Janet Baker of Dragon Systems delivering a speech that was completely "written" on a speech-driven word processor.

Although Baker says this technology is at the "Model T" stage, some analysts predict that speech recognition will eventually render computer keyboards obsolete.

Professor John Lavor, director of the university's Center for Speech Technology Research, feels that advances in speech technology will have the same impact on society as the invention of mechanical printing and that these advances will affect the whole social structure by the end of the century.

Other demonstrations at the speech technology conference included a telephone call box operated entirely by voice command and a system under joint Scottish/Japanese development to translate telephone conversations automatically.

For further information, contact Gregory F. Romano, Rifkind Pondel & Parsons, 11601 Wilshire Blvd., Los Angeles, CA 90025.

Current literature

The *VME Data Book 1988* from Force Computers includes the new VME/Plus line of 32-bit products. Force documents the technical features of each VME/Plus CPU (and related boards) and previews products presently in the design cycle for 1988 release. A companion 18-page, technical brochure discusses the company's 132-pin and 280-pin VME/Plus gate arrays, accompanied by reproducible system block diagrams for each CPU board.

The data book contains a product selection matrix that helps differentiate similar products. Free copies can be obtained via letterhead requests to the company (note European address) or from company representatives.

Force Computers, Inc., 3165 Winchester Blvd., Campbell, CA 95008; (408) 370-6300; or Force Computers GmbH, Daimlerstrasse 9, D-8012, Ottobrunn, West Germany; (089) 600-910; 560 pp.

If you can chew gum and walk on sidewalks at the same time, or handle a screwdriver while you watch TV, Wolfer Productions promises that you can assemble your own PC in two hours—or your money back.

Wolfer's how-to videocassette explains each step for assembling an IBM XT-compatible computer from a kit, showing components before assembly, actual installations, and close-ups of each connection. The package includes a list of kit suppliers with snap-on boards; you're on your own for the screwdriver.

Wolfer Productions, Computer Video Department, 10153½ Riverside Drive #153, Toluca Lake, CA 91602; $39.

If you're interested in automating the design and manufacture of integrated circuits, Technical Insights Inc. offers *Silicon Compilers: The Technology, The Vendors, The Opportunities*. The report provides histories of CAE, ASICs, gate arrays, and standard cells. It discusses silicon compilation: its uses, advantages, costs, and potential. The study also examines the role of silicon compiler generators and the way they are changing the silicon foundry business. Market analysis, forecasts, and technical data sheets from vendors are presented. The report also explores how to manage chip design capabilities and pursue business opportunities.

Technical Insights Inc., Dept. J90887, PO Box 1304, Ft. Lee, NJ 07024; 193 pp.; $950; overseas buyers add $35.

What's your PC IQ? Try Computer Trivia, a board game (as in regular, old board game) for those who either know "all about" personal computing or would like to learn. The game contains over 2000 questions in five categories: hardware, software, people and companies, history, and potpourri. A novice section trains beginners.

Orange Alps, Inc., 3030 First Avenue N.E., Cedar Rapids, IA 52402; (319) 364-8383; $29.95.

If you were unable to attend the IEEE Fifth Biennial Careers Conference in October 1987, it's not too late to obtain a copy of the proceedings. *The Engineer's Life and Career in Today's World* addresses the issues of career development in a period of rapid, severe industrial change; the balance between personal life and career; how to gain professional power; and the ethical questions raised by the Challenger and Chernobyl tragedies, among other topics.

IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08855; IEEE Catalog No. UHO 176-8; 164 pp.; $20 for IEEE or Computer Society members; $25 for nonmembers; add $4 for postage and handling.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 183 Medium 184 High 185

New Products

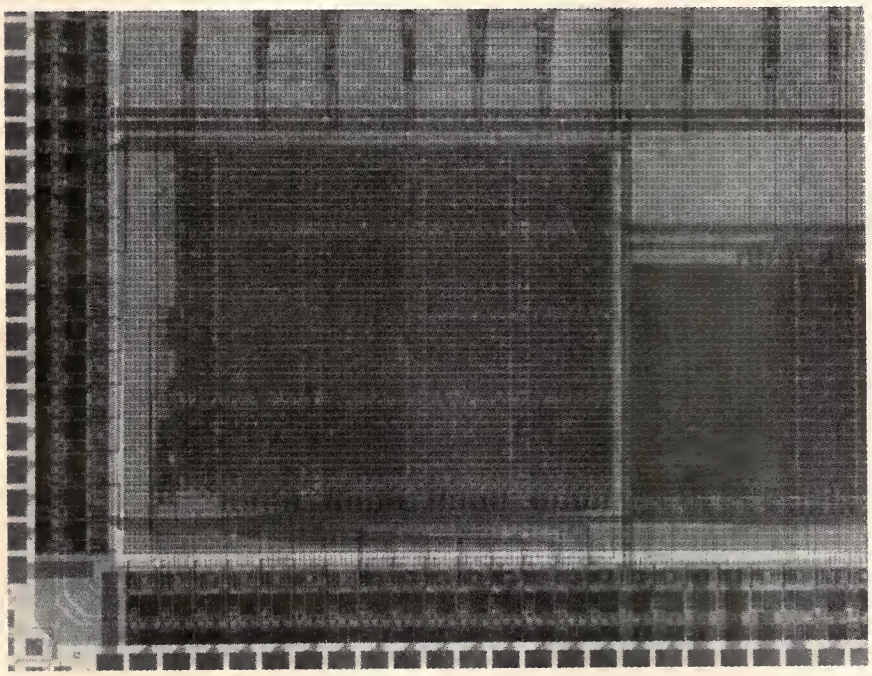
Marlin H. Mickle
University of Pittsburgh

Send announcements of new microcomputer and microprocessor products, and products for review, to Managing Editor, IEEE Micro, *10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578.*

Design capabilities soar

Noteworthy improvements in capabilities for design offer opportunities to use fewer components, reduce board space and system size, lower weights, and increase system reliability. The results for system vendors and end users include lower manufacturing and maintenance costs, greater manufacturing productivity, and because smaller power supplies are needed, more portable systems.

Complex gate array circuits achievable



Channel-Free architecture in the LCA100K Compacted Array Plus series helped implement the memory shown here. The block in the lower left corner is a 128 × 8 static RAM, and to its right is a portion of a 512 × 16 ROM.

The LCA100K Compacted Array Plus family uses LSI Logic's proprietary 0.7-micrometer channel-length HCMOS process. The series obtains usable gate counts of 100,000 gates on one chip. One chip built with the Channel-Free array technology can contain 16K static RAM, 64K ROM, and 46K usable logic gates—with gate delays of 460 picoseconds through a 2-input NAND. Modular Design Environment software manages the design of the complex chips. Real-time processing applications include digital signal processing, image processing, artificial intelligence, and speech recognition and synthesis. **LSI Logic Corporation.**

Reader Service Number 10

Integrate analog/digital functions on one chip

National Semiconductor Corporation's standard cell library now includes eight analog cells. These cells allow systems designers to combine analog and digital functions on one IC. The analog functions are two comparators, three operational amplifiers, a voltage reference, an analog switch, and resistors. Based on the advanced CMOS process, the functions operate with 5-volt power supplies. **National Semiconductor Corporation.**

Reader Service Number 11

CAE kit permits design of 50,000 gates

Daisy CAE workstations can now access the Portable Library design kit to produce array designs of 50,000 usable gates. Consisting of 240 element icons and four timing modules—one each for 2-micrometer and 1.5-micrometer gate arrays and standard cells—the kit can simulate multiple ASICs at the system level. **VLSI Technology, Inc.**

Reader Service Number 12

Reprogrammable, 20-pin, CMOS PALs announced

Designated TICPAL16R4/R6/R8/L8-55 are four, virtually zero-standby-power devices. The programmable array logic devices are compatible with TTL and CMOS logic and program in TTL levels. The series has a clock-to-Q time of 22 ns with a maximum propagation delay of 55 ns. The PALs are available in windowed, 20-pin, ceramic dual-in-line packages; nonwindowed, one-time programmable plastic DIPS are expected this spring. **Texas Instruments Incorporated; $5 each (1000's).**

Reader Service Number 13

Faster, lower-cost ways to mainframe performance

One-chip devices, intricate boards, and low-cost microcomputers offer increasingly faster speeds and complex capabilities. We are seeing products that meet the demands of users for highly integrated, versatile, high-performance systems to replace larger, more power-demanding systems.

DRAM controllers eliminate processor wait states

A family of integrated address controllers for use between processors and 16K, 64K, 256K, and 1M-bit dynamic RAMs is intended to be used with high-performance memory systems. The SN74ALSxxxx devices are fabricated in bipolar Impact technology to give each function an 18-ns address-to-Q performance and an Icc of 200 milliamperes. Each device contains functions for address multiplexing and refreshing. Timing control executes off chip through user-defined FPL devices. Military and commercial versions are packaged in plastic and side-brazed CERDIPs. **Texas Instruments Incorporated; $15 and $18 (1000's).**

Reader Service Number 14

Multiuser computer protected from system failures

The Parallel 400XR is an entry-level computer system that makes fault-tolerant computer systems available to small/medium-size companies. The series with Unix operating system supports 120 serial terminals and printers and uses microprocessors based on the MC68020. According to the company, the fault-tolerant design provides users with redundant technology that results in an estimated MTBF of more than 20 years. Self-diagnostic features provide continuous, automatic monitoring of system components, sending fail messages to the user and activating the redundant component or subsystem. **Parallel Computers; $42,900 to $80,000.**

Reader Service Number 15

GET UP AND RUNNING

Every issue of *IEEE Software* contains new ideas and practical research to drive your hardware. Look to us for information on operating systems, prototyping, parallel processing, compiling environments, software development, transputing, and distributed processing.

Name (please print)

Address

City State ZIP

2/88 Micro

Rates (1 yr., 6 issues):
☐ **$17** (members of an IEEE society)*
_____Mem. No.
☐ **$25** (members of other organizations)
_____Org.
_____Mem. No.

**IEEE society members: Pay $8.50 for three issues (half-year subscription) for orders submitted March-August.*

Enclose a check and mail to: *IEEE Software*, Circulation Dept., 10662 Los Vaqueros Cir., Los Alamitos, CA 90720.

IEEE Software

LCD driver resides on chip

The MC68HC05L6 contains an on-chip liquid crystal display driver, oscillator, CPU, 6208-byte ROM, 176-byte RAM, Serial Peripheral Interface, 16-bit timer, and audio tone generator. The LCD drive uses ⅔, ⅓ VLL divider circuitry up to 96 segments. Right now, this 8-bit microcomputer is available in die form only. **Motorola Microprocessor Products Group; $12.95 each (1000's) with $5300 mask charge.**

Reader Service Number 16

Chip accommodates 16-bit microcontroller

A 12-MHz CHMOS-III device called the 80C196KA controls computer peripherals, automotive, and industrial automation applications in real time. Features of the PLCC controller include a two-phase clock frequency and a register file of 232 bytes, permitting a three-operand multiple to perform two loads, one multiply, and a 32-bit store in 2.33 μs. On-chip peripheral functions include 256 bytes of RAM, a pulse-width modulator, a 10-bit A/D converter, an I/O unit, a serial I/O port, and 5 × 8 parallel I/O ports. **Intel Corporation; $18.75 (10,000's).**

Reader Service Number 17

Program EEPROMs remotely

Two 16K CMOS EEPROMs, the IDT78C16 and the IDT78C18, feature a 55-ns read access time and the Serial Protocol Channel for serial access. SPC allows rewrites to occur serially and offers remote programming capabilities. Applications for the CERDIP and LCC devices include industrial process control, military and commercial aircraft, programmable robotics, and intelligent telephones and PABXs. **Integrated Device Technology, Inc.; $15 and up (100's).**

Reader Service Number 18

XTs upgraded with OS/2, memory board

A replacement system board for IBM PC XTs and compatibles gives users access to 16M bytes of memory and OS/2 capability. The Xformer/286 comes with a 10-MHz 80286 processor, zero wait states to system-board memory, and 512K bytes of memory expandable to 1M byte on the board. **AST Research; $845.**

Reader Service Number 19

VME board controls seven SCSI targets

A VMEbus/SCSI host bus adapter controls up to seven SCSI targets with configurations set by user software. With optional daughter boards, the 720 can control 14 SCSI ports and two floppy drives. Combining the company's 8K-byte FIFO architecture and Dynathrottle DMA control, the adapter offers synchronous transfer rates of 4M bytes/s, asynchronous rates of 1.5M bytes/s, and DMA speeds of up to 18M bytes/s. Use of magnetic disks, optical disks, magnetic tapes, floppy drives, printers, and computer-to-computer communication is possible with the 720. **Xylogics, Inc.; $1795 and up.**

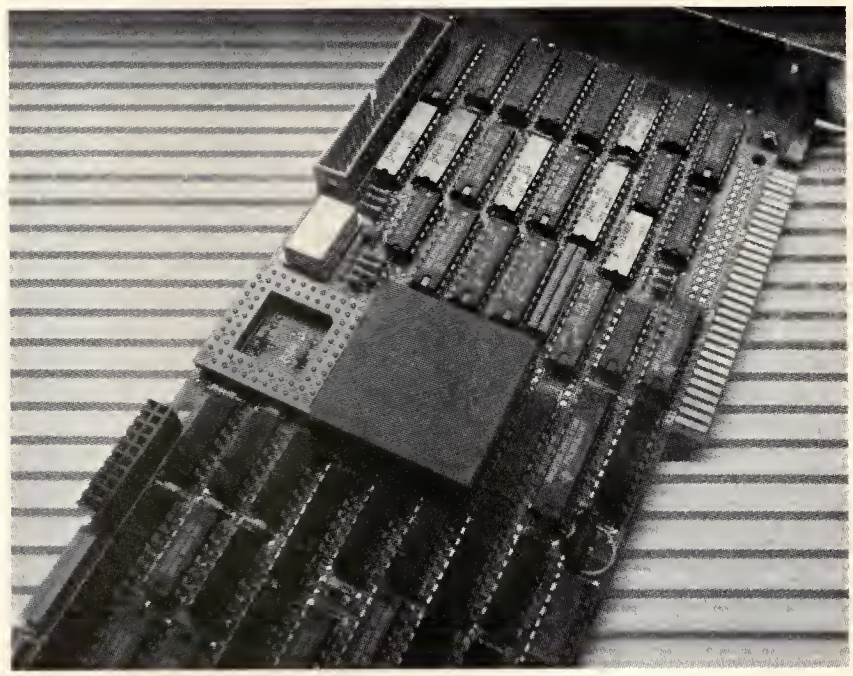
Reader Service Number 20

Supermicros accept 24 or 32 users

TS-386 super-microcomputers are based on the 32-bit 80386 microprocessor and operate at 16 MHz. The PC-compatible machines are available in 24-user and 32-user versions, bundled with the company's Basic operating system, tape backup and multiport expansion boards. Maximum disk size for all TS-386 systems is 230M bytes. **Thoroughbred Division, Concept Omega Corporation.**

Reader Service Number 21

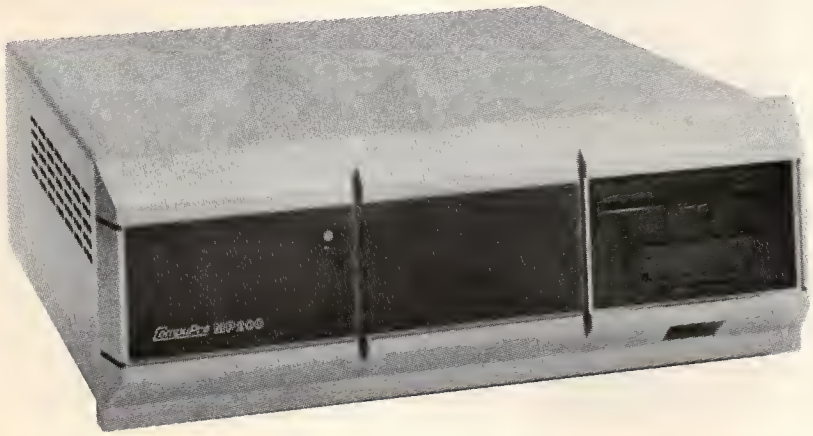
Add-in promises tenfold PC-performance increase



The Inboard 386/PC has no switches or jumpers to set and uses a standard ribbon cable to attach the board to the system board.

Inboard 386/PC is an 80386-based add-in board with 1M-byte, 32-bit memory. Tests run with Turbo Pascal compile programs (35,000 bytes/1272 lines) performed in two seconds with the board and 16 seconds without it. Tests on a dBase III index of 3200 records performed in 34 seconds with the board and 209 seconds without it. Inboard 386/PC accommodates an optional 2M-byte Piggyback Memory board. **Intel Corporation Personal Computer Enhancement Operation; $995.**

Reader Service Number 23



A parallel processing multiuser system, the MP 200 series, incorporates eight 16-bit processors, accommodating 16 users, application software, and data communications options. Features include the Concurrent DOS XM 5.2 operating system, a modular architecture, cache memory, direct connection to IBM PC, XT, AT, and PS/2 workstations, and system utilities for electronic mail and networked PCs. CompuPro; $13,700 to $19,000, depending on configuration.

Reader Service Number 22

Mac286 lets Macintosh II run MS-DOS applications

The Mac286 system allows Apple Macintosh II users to run MS-DOS applications at speeds comparable to an IBM PC AT. The MS-DOS programs appear in a window on the screen, and users can move between DOS programs and Macintosh programs with a mouse. Data can be exchanged and shared between the two environments. Mac286 contains an 8-MHz 80286 microprocessor, 1M-byte RAM, and a socket for an optional 80287 math coprocessor. A 5.25-inch or 3.5-inch external floppy-disk drive can be attached for access to MS-DOS disk media. **AST Research; $1499.**

Reader Service Number 24

High-volume storage devices satisfy sophisticated users

Sophisticated data, graphics, and imaging systems demand new ways to store and access large amounts of information. Alternative storage technology for microcomputers offers a variety of capabilities: on-line access, management systems, and fast access times. Some of the latest offerings summarized here reveal the extent of changes in mass storage products.

Disk subsystem features transparent DOS

The internally or externally mounted ISi 525 WC optical disk subsystem includes WORM-TOS optical software for transparent PC/MS-DOS applications and utilities. The ISi 525 drives use removable SuperStore 2000 5.25-inch cartridges in 115M-byte and 230M-byte formats. DOS features include subdirectory trees, file management and selection, direct access to the optical disk, and file tracing. **Information Storage Inc.; $2595 (internal mount), $2795 (external mount).**

Internal **Reader Service Number 25**
External **Reader Service Number 26**

Device accesses display databases

With a smart-disk controller and four to 16 standard disk drives connected in parallel, the ImageDisk storage subsystem supports the Ramtek 4660 imaging display system. ImageDisk stores from 560M bytes to 16G bytes of information and transfers data bursts at 8M bytes/s. A 1.2G-byte subsystem can store 4800, 8-bit, color, 512×512-resolution display frames, detecting and handling errors. **Ramtek Corporation; $65,000.**

Reader Service Number 27

Drive subsystem adds operating system

A WORM optical disk system called Model 525 WORM now provides the transparent WORM-TOS operating system to support files of 123M bytes in size. WORM-TOS enables users to treat N/Hance and ISI WORM disk systems as if they were magnetic disks while using 18K bytes of main memory. The DOS-compatible system supports multiple logical partitions and allows users to write to the optical disk directly from application programs through normal DOS calls. **N/Hance Systems; $235.**

Reader Service Number 28

WORM unit manages 66 disk cartridges

Optofile automatically manages up to 66 optical, 400M-byte disk cartridges and up to four 5.25-inch, write-once-read-many (WORM) drives. With an SCSI interface the 26.4G-byte, on-line storage system selects a disk, turns it right side up, and inserts it into a drive. Two or more Optofiles can be daisy-chained using RS-232 connections. The desktop system in various configurations serves Sun, MicroVAX II, PC, PS/2, and Macintosh computers. **Optotech, Inc.; $9950, entry-level system.**

Reader Service Number 31

WORM drive promises 65-ms access time

The LaserBank 800 optical drive offers 800M bytes of storage, a 65-ms access time, and DOS throughput at 200 kilobytes/s. An SCSI card allows users to connect the drive with PCs and compatibles and supports PC Network, PC Token Ring, 3Com, and Ethernet networks. FlashBack software retrieves file versions previously written to the LaserBank 800. Users can access seven logical drives and partition each optical disk cartridge into 16M- to 400M-byte volumes. **Micro Design International; $9995.**

Reader Service Number 29

Drive exchanges Sun/VAX/PC/Macintosh data

Laser DataBank WORM optical drive subsystems contain proprietary software that permits data to be exchanged among Sun, VAX, PC, and Apple computers. The 5.25-inch subsystems include a 400M-byte disk drive capable of storing CAD, CAE, CAM, computer-aided publishing, and other data/graphic-intensive applications. The externally mounted Laser DataBank also contains a read-write device driver, a PC controller, and an SCSI controller that connects four drives simultaneously for 800M bytes of on-line storage. **Optotech, Inc.; $2995 to $6950, depending on version.**

Reader Service Number 30



A 4G-byte, 12-inch, write-once optical disk drive, the WM-S500, transfers data at 8M bit/s and an average seek time of 150 ms. Designed for real-time, on-line access of large archival storage, the stand-alone WM-S500 provides an SCSI interface and resides in a 19-inch RETMA rack. Toshiba America Inc.; $11,495; OEM pricing available.

Reader Service Number 32

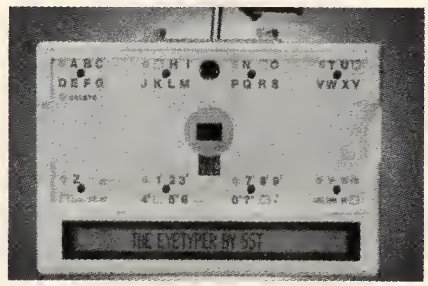
Handicapped persons benefit from recent products

Input and output devices continue to accommodate a variety of users working in vastly different environments. Some are designed specifically for the disabled, returning a degree of control to lives restricted by physical limitations. Others just make life easier for all of us. Here's a sampling of a few particularly appealing devices.

Nonspeakers can "talk" by moving their eyes

EyeTyper 300 measures the reflection of an infrared light on the cornea of a user's eye, allowing the user to select a letter, word, phrase, or picture for display. The display appears in large print in 40-character, single lines and in synthesized speech. EyeTyper can be connected to computers, printers, or an environmental control device. **Sentient Systems Technology, Inc.**

Reader Service Number 33



The EyeTyper 300 eye-gaze-controlled keyboard system allows speeds of 10 to 12 words per minute.

Infrared touch panel creates light beams on display

You can bring touch control to 12- and 13-inch screens with the Addressable Touch Panel, which creates an invisible, 40 × 24 (x,y) matrix of light beams in front of the display. The host CPU carries out addressing and scanning of the panel to determine the finger/stylus locations. The touch panel fits between the CRT and enclosure. **Electro Mechanical Systems; $98 (10,000's) plus nominal NRE charge for OEMs.**

Reader Service Number 34

A/D converters offer MIL screening

Two 16-bit A/D converters targeted for size-sensitive military/industrial applications offer 17-μs maximum conversion times. The DIP-packaged MN5295 and MN5296 feature ±0.003 percent FSR maximum linearity error, six user-selectable input ranges, serial and parallel output data formats, and 1.2-W maximum power consumption. High-reliability screening per MIL-STD-883 is optional on some models. **Micro Networks/Unitrode; $162 (100's) starting price.**

Reader Service Number 35

Utility prevents time loss from disasters

The IBM PC-DOS Disk Watcher is a RAM-resident utility that addresses disaster prevention needs. It automatically pops up on screens when a disk is full, allowing users to copy, move, erase, or back up files to diskettes. Another feature traps a full-disk condition when data is imported via communications packages. **RG Software Systems, Inc.; $79.95.**

Reader Service Number 36

Drafting plotters adjust to various media

Designed for applications requiring fast pen speed, acceleration, and throughput, the DMP-60 drafting plotters perform with a variety of English and metric media formats. The single-pen plotters offer adjustable media size capabilities and accept numerous add-on performance options such as multipen accessories, optical scanners, expanded buffer boards, and a Kanji character set board. The DMP-61 has an axial pen speed of 32 ips and an axial acceleration of 4 g. The DMP-62 handles 23 media sizes from 8.5 × 11 inches to 36 × 48 inches with a speed of 24 ips. **Houston Instrument, Ametek Division; $4695 (DMP-61), $6495 (DMP-62).**

Reader Service Number 37



With the Bigmouth digital recording system, your PC or compatible can integrate voice with programs to create programs with sound effects, talking demos, tutorials, and custom telephone management functions. Bigmouth contains a plug-in half card, software, external speaker, phone cables, and documentation. Talking Technology, Inc.; $239.

Reader Service Number 38

Turn a touch-tone telephone into a terminal

TeleCommand allows touch-tone telephones to be used in a two-way phone conversation with IBM PCs and compatible computers. The intelligent voice technology unit converts touch tones into keystrokes available on a keyboard, maintaining security precautions and reading back to callers screen-displayed text in synthesized or digitized voice. Additional features are outbound calling, call forwarding, and call conferencing. **Computer Solutions International; $1495.**

Reader Service Number 39

PS/2 mouse supports VGA graphics

PC Mouse PS/2 includes driver support for VGA graphic modes and includes Designer Pop-up Menus for business applications. The mouse uses light to measure movement rather than the rotating ball common to a mechanical mouse. One version plugs into IBM PS/2 serial ports, and a Bus Plus version fits the Model 30. **MSC Technologies, Inc.; $159.**

Reader Service Number 40

Non-IBM processors can use IBM E-mail

Non-IBM electronic mail systems can exchange messages transparently with Systems Network Architecture Distribution Services using the Orion SNADS Facility. The C software provides store-and-forward switch capabilities on a SNADS network to send and receive documents and distribute bit streams of electronic messages, files, graphics, and voice. Featured is a generic interface to directory services. The SNADS Facility can be installed in processors with an implementation of the IBM SNA LU6.2 peer-to-peer communication protocol. **Orion Network Systems, Inc.**

Reader Service Number 41

SIMD system develops GAPP-based software

The NCR45SPDS is a single-instruction, multiple-data system, which supports the development of operational and algorithmic software for Geometric Arithmetic Parallel Processors. The menu-driven SPDS system lets users access one of four different operational modes. System software runs on an NCR PC8 or other IBM PC AT-compatible computer and contains an NCR-GAL compiler and a GAPP microcode generator. Hardware includes one to four GAPP array cards, an SIMD controller card, host interface card, and optional frame grabber. **NCR Corporation; $28,500 (one-array board configuration).**

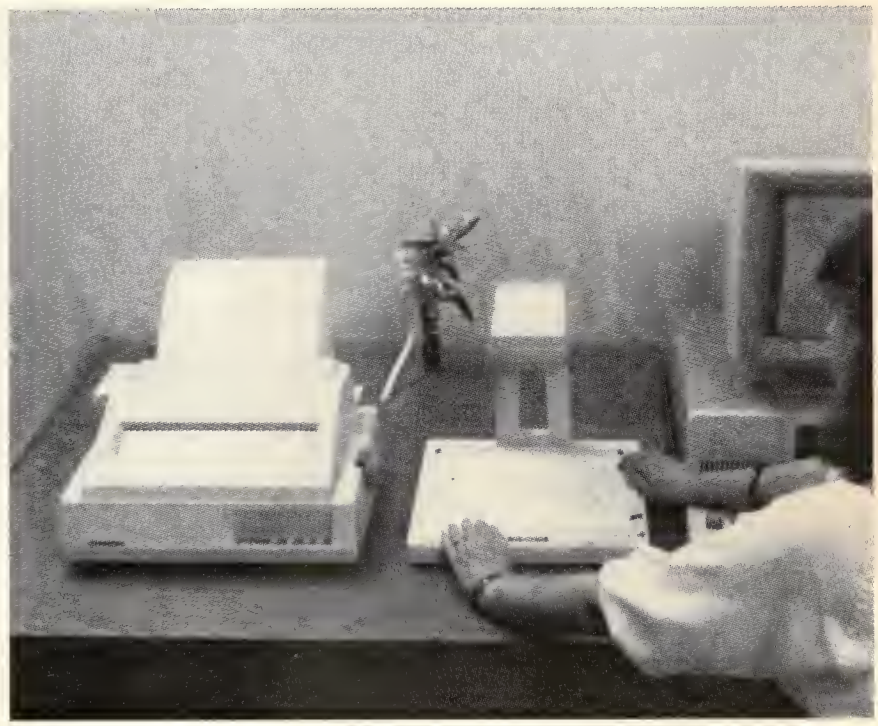
Reader Service Number 42

Math package aids 386 PCs with Weitek coprocessors

An interactive scientific and engineering software package, MATLAB, integrates numerical analysis, matrix computation, signal processing, and graphics on 80386-based personal computers. A second version called 386/Weitek-MATLAB supports the Compaq DeskPro 386/20 and compatibles equipped with Weitek's 1167 math coprocessor. Mainframe-sized data sets can be used by accessing a protected mode. **The Math Works Inc.; $1495 (386-MATLAB), $1995 (386/Weitek-MATLAB.**

386 **Reader Service Number 43**
386/Weitek **Reader Service Number 44**

Overhead scanner feeds PCs/Macintoshes



To operate the Image Scanner, users place the document to be scanned face up on the surface; the overhead optics scan the document and capture the data as it is scanned. The software reads MacWrite and Microsoft Word and offers paint and draw capabilities.

The 200-dpi Model N-205 scanner is a stand-alone peripheral and a bundled hardware and software desktop publishing system. A proprietary overhead scanning design lets the device operate in ambient lighting environments with IBM PCs and compatibles and Macintosh computers without interface add-in hardware. Users can input page layout images, text, or documents, calling up an image on the terminal, editing or manipulating it and printing it. **Chinon America Inc., Information Equipment Division; $695 (stand-alone version).**

Reader Service Number 46

Macintosh package retouches 64 gray levels

Macintosh-based image processing software lets you manipulate the gray-level information of images generated by high-resolution scanners. ImageStudio crops, silhouettes, rotates, and moves image elements and changes their brightness and contrast. Design tools emulating a paintbrush, water drop, charcoal, and fingertip permit effects such as air brushing, edge softening, and paint smearing. **Letraset USA.**

Reader Service Number 45

Text, graphics, barcodes print at 10 pages/minute

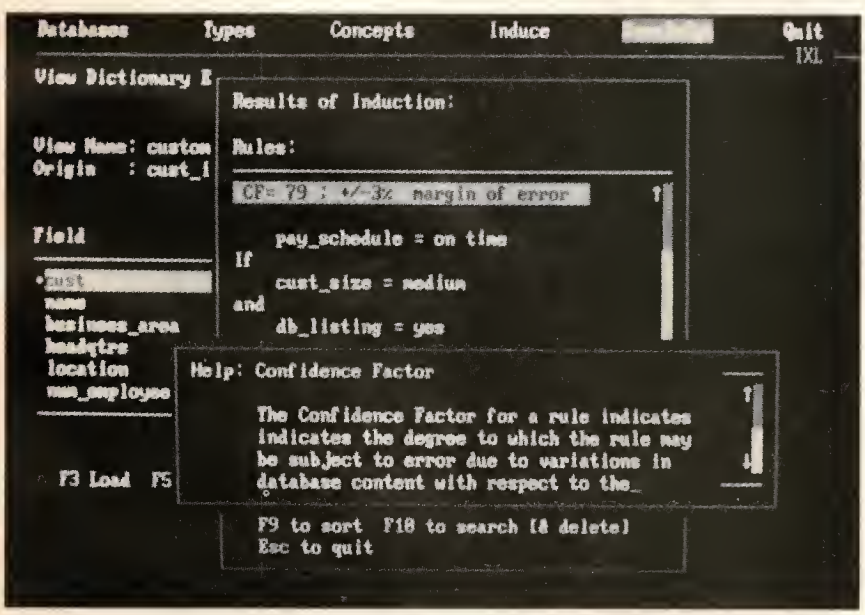
The F-1000A 10-ppm, desktop laser printer can produce text, graphics, and 39 barcode styles at 300-dpi resolution. Emulator of seven printers, the F-1000A with 512K RAM has 79 resident, bit-mapped fonts with eight foreign language character sets, and three user-modifiable Dynamic fonts for typestyle creativity. Two card slots accept customized IC cards that can store personalized logos, business forms, fonts, and signatures. **Kyocera Unison, Inc.; $2895.**

Reader Service Number 47



The IDR color image processing system displays multiple still video images together with graphics and text on an RGB analog television monitor (above). The heart of the system is the CMD capture, merge, and display board consisting of three 82786 graphic coprocessor chips; an on-board, time-based correction circuit; and an A/D and D/A converter. Users can freeze and store a video frame at random, enhance its size and color, and overlay images on a screen while live video is displayed (below). IDR Inc.

Reader Service Number 48



IXL: the Machine Learning System uses both statistical and artificial intelligence techniques to automatically discover knowledge hidden in large databases. IXL reads databases in dBase III, Lotus, and ASCII format; deals with inexact and omitted data; and allows users to specify the level of acceptance error. The expert system environment requires an IBM PC XT or AT with 512K RAM and a hard disk. Intelligence Ware Inc.; $490.

Reader Service Number 49

Software emulates math coprocessor

The EMU287 math coprocessor software emulator resides in 8000 bytes of code space on Intel 80xx processors. With a 5-MHz 8086, the emulator offers adds of 580 μs (typical) and 760 μs (maximum). With an 8-MHz 80286, typical values of adds are 175 μs and maximum timings are 250 μs. **United States Software Corporation; $2500.**

Reader Service Number 50

SCSI disk drive offers PC users 10M-bps transfers

A 5.25-inch Small Computer Systems Interface disk drive helps PC users and integrators upgrade hard-disk performance and capacity. PC PAK-Performance Advantage Kit offers formatted capacities of 70, 90, 105, 125, and 145 megabytes. Data transfers at 10M bps; the average access time is 23 ms. PC PAK's host adapter supports two floppy-disk drives, six hard drives, and a backup SCSI tape drive. **Micropolis Corporation; $2495.**

Reader Service Number 51

LAN software shares PC, Macintosh files

Two Versions 2.0 of the TOPS Local Area Network software for transparent file sharing between IBM PC, Apple Macintosh, and Unix-based computers have been introduced. TOPS/DOS Version 2.0 allows PCs on the TOPS network to access dedicated printers formerly available to only one user. It also offers FlashTalk, the PC-to-PC communications architecture that operates on TOPS at three times its former AppleTalk speed.

TOPS/Macintosh introduces a Remember function that lets users automatically make files available to the network and access remote files. It is compatible with Apple File Protocol applications. Both of the new versions support AppleTalk zones and offer improved password protection. **TOPS; installed on all new TOPS; upgrades free or $29 per node, depending on purchase date.**

Reader Service Number 52

LAN transceiver contains rotary vampire tap block

A 2.4 × 1.7 × 0.95-inch LAN transceiver with a rotary vampire tap block permits T-perpendicular branching or H-parallel branching. An invasive tapping system permits later-movable Ethernet connections. According to the company, MTBF specification at 30°C is 1,325,000 hours. **Fujikura America, Inc.**

Reader Service Number 54

Telephone switcher avoids FAX access codes

AutoSwitch TF 4 permits remote access and control of three different telephone devices connected to one telephone line. Especially designed for facsimile machines, the TF 4 incorporates a circuit to automatically recognize an incoming facsimile carrier signal and directs the call immmediately to the receiving machine. A typical installation can include an answering machine, facsimile machine, and a computer modem. **Command Communications, Inc.**

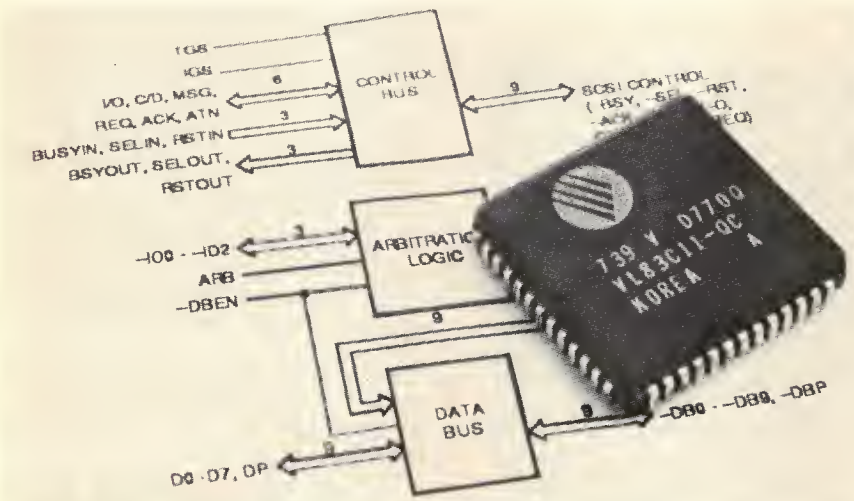
Reader Service Number 53

Chips support PC designers

Systems designers can achieve design continuity for 80286 system configurations with Zymos-designed Multifunction Peripherals. The two-chip core logic set offers users a VLSI solution for PC AT-compatible computers of varying speeds and wait-state combinations. The 82231 10- and the 82230 12-MHz development kits come with evaluation board, schematics, technical documentation, and circuit board films. **Intel Corporation; $60 (10 MHz); $90 (12 MHz) (100's).**

Reader Service Number 55

CMOS SCSI transceiver offered

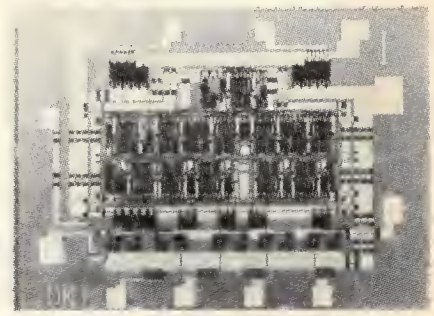


According to the company, the CMOS VL83C11 operates at less than one tenth the power required for its NMOS NCR 8310 equivalent (excluding interface current). It meets the electrical specifications of SCSI for system cable lengths up to six meters.

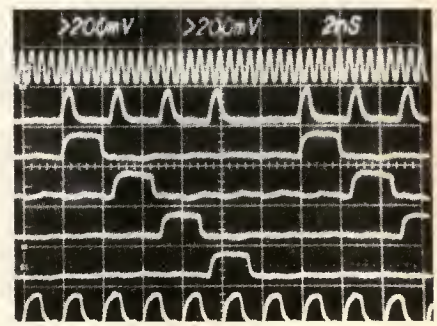
The VL83C11 General-Purpose Bus Transceiver Chip performs the function of a Small Computer Systems Interface transceiver. The two-micrometer CMOS device serves as a 48-mA bus transceiver chip for all SCSI bus signals, incorporating high-current, single-ended drivers for the SCSI bus. The VL83C11 interfaces directly to the future VL53C86 or NCR 53C86 SCSI Protocol Controller families. The 52-pin, PLCC-packaged VL83C11 is functionally equivalent to the NMOS NCR 8320. **VLSI Technology, Inc.; $8.13 (1000's).**

Reader Service Number 56

GaAs telecom devices operate at 2G bits/s



(a)



(b)

Typical power consumption for the 4:1 GaAs multiplexer is 400 mW (a); samples of waveforms produced by the 1:4 GaAs demultiplexer, which consumes 850 mW (b).

Proprietary gallium arsenide multiplexer and demultiplexer ICs operate at speeds up to 2 gigabits/second in optical transmission equipment without degradation when connected to a heavy load. Designed with super-buffer field-effect transistor logic (SBFL), the flat-pack, two-chip sets include 2:1 and 4:1 multiplexers and 1:2 and 1:4 demultiplexers. Their scale in terms of equivalent gates ranges from 80 to 200. Sample quantities are planned for this spring. **Oki Semiconductor; $1250.**

Reader Service Number 57

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

Low 180 Medium 181 High 182

Product Summary

Marlin H. Mickle
University of Pittsburgh

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.

Manufacturer	Model	Comments	Rs No.
Systems			
BBN Advanced Computers	Butterfly Ada parallel computers	Verdix Ada Development System source code ports Ada language to the general-purpose Butterfly GP1000 parallel processing systems. VADS, a production-quality compiler, complies with ANSI/MIL-STD-1815A. Included are a symbolic debugger, library maintenance utilities, diagnostics, and a runtime system for tasking, exception handling, and I/O. A 30-processor, 75-MIPS GP1000 with Butterfly Ada sells for $500,000.	80
Grid Systems	GridCase 1500 series; GridLite Plus	Portable laptop series offers 12-lb. models based on 80C286 (Model 1520) and 80C386 (Model 1530) microprocessors with software development tools and three external portable disk drives. The 9-lb., 10M-byte, hard-disk GridLite Plus includes an 80C86, 640 × 200 supertwist backlit LCD, 640K-byte RAM, 1.4M-byte internal disk drive, and 1M-byte ROM with 8 ROM slots. Prices start at $1950.	81
Software			
Alsys, Inc.	Ada Cross-Development System	Software system supports the development of embedded and real-time Ada applications on bare-board configurations of the iAPX86 microprocessor family with no underlying operating system support. Hosted on an IBM PC AT or compatible under DOS 3.0, the cross-development system requires 512K bytes of main memory and comes bundled with a 4M-byte memory board for the host. Programmers can address 16M bytes of memory, if available, with protected-mode applications.	83
Index Technology	Excelerator systems	Analysis and design software, designed for Sun Microsystems workstations, provides a Snapshot feature so users can display data in multiple windows while simultaneously performing other tasks. Excelerator serves CASE products; Excelerator/RTS lets engineers design complex real-time systems with time and control processing requirements. $8400 each version; discounts with two copies.	84
TriVector	Tiger graphics/ database translator	Proprietary software database and format translator transmit digital graphics data between textual and graphic systems. Output can be obtained in one or more IGES subsets; custom, internal non-IGES formats; or Postscript, HPGL, and Calcomp formats. Platforms supported are Apollo, Hewlett Packard, Sun, and the IBM PC AT and compatibles.	85
Peripherals			
Brother International	M-2518 printer	Data processing printer outputs on thick, multipart forms at draft speeds of 360 cps. A Paper Express feature aligns the printing on each copy of invoices and statements of up to six parts. The Epson- and IBM-compatible system prints near-letter quality at 75 cps and letter quality at 50 cps. $1295.	86
Genoa Systems	Galaxy/MC tape backup systems	IBM Micro Channel systems provide tape backup for file transfer from other PCs to the Personal System/2 models. The 60M-byte data cassette model and 60M- or 125M-byte cartridges provide high-speed image backups at 5M bytes/minute, restore individual files from image, and support Novell Advanced NetWare and 3COM Etherseries LANs. $1250 to $2500.	87

Advertiser/Product Index

FOR DISPLAY ADVERTISING INFORMATION CONTACT:

Northern California and Pacific Northwest: Don Farris Co., 161 W. 25th Ave., Ste. 102B, San Mateo, CA 94403; (415) 349-2222.
Jack Vance, P.O. Box 3205, Saratoga, CA 95070.
Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Ste. 100, Torrance, CA 90505; (213) 373-9604.
Southwest: The House Co., 3000 Weslayan, Ste. 345, Houston, TX 77027; (713) 622-2868.
Midwest: Robert Acker & Associates, 480 Central Ave., Northfield, IL 60093; (312) 441-6050.
East Coast: Hart Associates, PO Box 339, 42 Lake Blvd., Matawan, NJ 07747; (201) 583-8500.
New England: Arpin Associates, PO Box 227, Weston, MA 02193; (617) 899-5613.
Europe: Heinz J. Gorgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G); (0 21 53) 8 99 88.

Advertising Director: Dawn Peck.
For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE Micro, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; (714) 821-8380.

Moving?

Name (Please Print) _____

New Address _____

City _____

State/Country _____ Zip _____

☐ Address change: Please notify us 4 weeks in advance

Mail to:
IEEE Micro, Circulation Dept.
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720-2578

ATTACH LABEL HERE

- This address change notice will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

	RS #	Page#
Boards		
Add-in board	23	81
Memory board	19	80
VME board	20	81
Chips		
A/D converter	35	83
Analog/digital	11	79
Array logic device	13	79
CMOS EEPROM	18	80
Controller	17	80
Design development kit	55	86
DRAM controller	14	80
GaAs multiplexer/demultiplexer	57	80
Gate array	10	79
Transceiver chip	56	86
Components		
LCD driver	16	80
Conference & Courses		
Systems design & networks	—	Cover 1V
WESCON/88	—	2-3
Data Communications		
Electronic mail	41	84
Graphics/database translator	85	87
LAN software	52	85
LAN transceiver	54	86
MS-DOS application	24	81
Telephone switcher	53	86
I/O Related Equipment		
Data processing printer	86	87
Drafting plotter	37	83
Eye typer	33	83
Image scanner	46	84
Infrared touch panel	34	83
Intelligent voice	39	83
Laser printer	47	84
PS/2 mouse	40	83
Memory/Storage Equipment		
Disk drive	51	85
Disk subsystem	25-26, 27	82
Optical disk	28	82
RAM-resident	36	83
Tape backup	87	87
WORM optical drive	29-32	82
Software		
Ada application	83	87
Analysis & design	84	87
Systems		
CAE workstation	12	79
Computer	15, 21, 80-81	80, 81, 87
Digital recording	38	83
Expert systems environment	49	85
Image processing	45, 48	84, 85
Parallel processing	22	81
SIMD system	42	84
Test & Measurement Equipment		
Math coprocessor emulator	50	85
Math package	43, 44	84

READER SERVICE CARD

IEEE MICRO

February 1988 issue
(card void after August 1988)

Name_____

Title_____

Company_____

Address_____

City_____ State___ ZIP_____

Country_____ Phone (____) _____

Please send
(Circle those you want):

201 Publications catalog
202 Membership information
203 Student membership information
204 *IEEE Micro* subscription information

Reader interest
(Circle what you liked, add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles/departments) **in the shaded section of this card under Product Inquiries.**

Product inquiries **1**
(circle the numbers for products and advertisers you want more information on)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

READER SERVICE CARD

IEEE MICRO

February 1988 issue
(card void after August 1988)

Name_____

Title_____

Company_____

Address_____

City_____ State___ ZIP_____

Country_____ Phone (____) _____

Please send
(Circle those you want):

201 Publications catalog
202 Membership information
203 Student membership information
204 *IEEE Micro* subscription information

Reader interest
(Circle what you liked, add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles/departments) **in the shaded section of this card under Product Inquiries.**

Product inquiries **2**
(circle the numbers for products and advertisers you want more information on)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

SUBSCRIPTION CARD

SUBSCRIBE TO IEEE MICRO

☐ YES, sign me up! ☐ Renew my subscription!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only **$17** for a year's subscription (6 issues). Subscriptions are annualized with membership. For orders submitted March through August, please pay half the given full-year fee for a half-year's subscription.

Society:_____ IEEE membership no:_____

☐ YES, sign me up! ☐ Renew my subscription!

If you are a member of ACM, NSPE, ACS, IEE (UK), SCS, IPSJ, IECEJ, or any other technical society, pay the sister society rate of only **$25** for a year's subscription (6 issues).

Society:_____ Mem. no. (if any):_____

Full Signature *Date*

Name

Street

City

State/Country *ZIP/Postal Code*

☐ Payment enclosed

☐ Charge to ☐ Visa ☐ MasterCard ☐ AmEx

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Charge Card Number

Mo. Yr.
☐☐ ☐☐
Exp. Date

M288

Charge orders also taken by phone:
(714) 821-8380 8:00 a.m. to 5:00 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader service inquiries, use other side

PLACE STAMP HERE

PO box address for reader service cards only

IEEE MICRO

Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

For reader service inquiries, use other side

PLACE STAMP HERE

PO box address for reader service cards only

IEEE MICRO

Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Computer Society of the IEEE
Circulation Dept.
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720-9970

THE COMPUTER SOCIETY

A member society of the Institute of Electrical and Electronics Engineers, Inc.

Executive Committee

President: Edward A. Parrish, Jr.*
Vanderbilt University
School of Engineering
Nashville, TN 37235
(615) 322-2762

President-Elect: Kenneth R. Anderson*

Vice Presidents
Education: Gerald L. Engle†
Technical Activities: Helen M. Wood (1st VP)*
Area Activities: Willis King†
Conferences and Tutorials: Joseph E. Urban (2nd VP)*
Membership and Information: Merlin G. Smith†
Publications: James H. Aylor†
Standards: Laurel Kaleda

Secretary: Duncan H. Lawrie
Treasurer: Barry W. Johnson†
Past President: Roy L. Russo*
Division V Director: Harriett Rigas†
Division VIII Director: H. Troy Nagle, Jr.†
Executive Director: T. Michael Elliott†
*voting member of the Board of Governors
†nonvoting member of the Board of Governors

Board of Governors

Term Ending 1988	Term Ending 1989
Mario Barbacci	Bill D. Carroll
Victor Basili	Lansing Hatfield
Paul L. Borrill	Duncan H. Lawrie
Lorraine M. Duvall	David Pessel
Michael Evangelist	Susan L. Rosenbaum
Allen L. Hankinson	Sallie V. Sheppard
Laurel Kaleda	Bruce D. Shriver
Ted Lewis	Harold S. Stone
Ming T. Liu	Akihiko Yamada
Earl E. Swartzlander, Jr.	Marshall C. Yovits

Next Board Meeting

8:30 a.m.-5 p.m., March 4,
Cathedral Hill Hotel, San Francisco

Senior Staff

Executive Director: T. Michael Elliott
Editor and Publisher: True Seaborn
Director, Computer Society Press: Chip G. Stockton
Director, Conferences: William R. Habingreither
Director, Finance and Administration: Mary Ellen Curto

Offices of the Computer Society

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Telex: 7108250437 IEEE COMPSO

Publications Office
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
Membership and General Information: (714) 821-8380
Publications Orders: (800) 272-6657

European Office
13, Avenue de l'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Telex: 25387 AVVALB

Purpose

The Computer Society strives to advance the theory and practice of computer science and engineering. It promotes the exchange of technical information among its 90,000 members around the world, and provides a wide range of services which are available to both members and nonmembers.

Membership

Members receive the highly acclaimed monthly magazine *Computer*, discounts on all society publications, discounts to attend conferences, and opportunities to serve in various capacities. Membership is open to members, associate members, and student members of the IEEE, and to non-IEEE members who qualify as affiliate members of the Computer Society.

Publications

Periodicals. The society publishes six magazines (*Computer, IEEE Computer Graphics and Applications, IEEE Design & Test of Computers, IEEE Expert, IEEE Micro, IEEE Software*) and three research publications (*IEEE Transactions on Computers, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Software Engineering*).

Conference Proceedings, Tutorial Texts, Standards Documents. The society publishes more than 100 new titles every year.

Computer. Received by all society members, *Computer* is an authoritative, easy-to-read monthly magazine containing tutorial, survey, and in-depth technical articles across the breadth of the computer field. Departments contain general and Computer Society news, conference coverage and calendar, interviews, new product and book reviews, etc.

All publications are available to members, nonmembers, libraries, and organizations.

Activities

Chapters. Over 100 regular and over 100 student chapters around the world provide the opportunity to interact with local colleagues, hear experts discuss technical issues, and serve the local professional community.

Technical Committees. Over 30 TCs provide the opportunity to interact with peers in technical specialty areas, receive newsletters, conduct conferences, tutorials, etc.

Standards Working Groups. Draft standards are written by over 60 SWGs in all areas of computer technology; after approval via vote, they become IEEE standards used throughout the industrial world.

Conferences/Educational Activities. The society holds about 100 conferences each year around the world and sponsors many educational activities, including computing sciences accreditation.

European Office

This office processes Computer Society membership applications and handles publication orders. Payments are accepted by cheques in Belgian francs, British pounds sterling, German marks, Swiss francs, or US dollars, or by American Express, Eurocard, MasterCard, or Visa credit cards.

Ombudsman

Members experiencing problems — late magazines, membership status problems, no answer to complaints — may write to the ombudsman at the Publications Office.

Information

Use the Reader Service Card to obtain the following material:

- Membership information and application (RS #202)
- Publications catalog (proceedings, tutorials, standards) (RS #201)
- Periodicals subscription application/information for individuals (members, sister-society members, others) (RS #200)
- Periodicals subscription application/information for organizations (libraries, companies, etc.) (RS #199)
- List of awards and award nomination forms (RS #198)
- Technical committee list and membership application (RS #197)
- Directory of officers, board members, committee chairs, representatives, staff, chapters, standards working groups, etc. (RS #196)

Systems Design & Networks Conference

The Conference on Computer Systems, Peripherals and Networks

April 19-21, 1988 — **Santa Clara Convention Center, Santa Clara, CA**

SDNC is the only locally supported conference with the cosponsorships of the local Chapter of the Computer Society, Santa Clara Valley Section of IEEE, and the Bay Area Council of IEEE. The technical committee of SDNC has put together an outstanding technical program in the areas of design automation, local area networks and mass storage systems.

SDNC 88 is comprised of technical sessions as well as tutorial courses. The technical sesisons will describe state-of-the-art in technology, architectures, software and applications. The attendee, whether a circuit designer, a systems designer or a networks specialist, will benefit from a variety of technical sessions and tutorial courses. Each day will focus on a specific topic. The following is a highlight of a highly exciting technical program.

REGISTER EARLY! SEATING IS LIMITED.

STATE-OF-THE-ART TECHNICAL SESSIONS

Tuesday, April 19
(8:30 — 5:30)

Integrating Tools for Design, Manufacturing and Test

- Design tools for chips, boards, and systems
- Systems integration methodology
- Computer Aided Software Engineering (CASE)
- Engineering Information System (EIS)
- Computer Integrated Manufacturing (CIM)
- Modern technologies in CIM
- Tools for managing engineering projects
- Specification languages and much more . . .

Wednesday, April 20
(8:30 — 5:30)

Putting LANs to Work

- User Network Interfaces
- Network Operating Systems
- LAN hardware-Ethernet, Token ring etc.
- LAN Standards
- LAN performance analysis and tradeoffs
- Network Connectivity
- Twisted Pair technologies and implementations
- FDDI and high performance LANs
- OS/2 integration in LAN environment
- Distributed file systems and much more . . .

Thursday, April 21
(8:30 — 5:30)

Mass Storage Trends and Systems Integration

- Emerging magnetic mass storage trends
- Thin film media thin film heads
- Parallel Architectures
- Vertical recording and embedded servos
- New Bubble and Tape subsystems
- High reliability systems
- Optical WORM & R/W systems
- Systems Interfaces SCSI, IPI, etc.
- Embedded Controllers and much more . . .

TUTORIALS

April 19	• Local Area Networks and Applications
April 20	• Magnetic and Optical Mass Storage Systems
April 21	• Computer Aided Software Engineering (CASE) and Applications

EXHIBITS

- Design tools, hardware software implementations
- LAN products, off-the-shelf hardware, software
- Magnetic, optical disk drives, controllers, software

FOR MORE INFORMATION CONTACT KEN MAJITHIA, CHAIRMAN 408-256-4914 (day)
1248 Olive Branch Lane, San Jose, CA 94120. 408-997-2755 (eve)